

AD-A056 660 MOORE SCHOOL OF ELECTRICAL ENGINEERING PHILADELPHIA P--ETC F/6 9/3
AUTOMATED TEST DESIGN. (U)
JUN 78 C TINAZTEPE

UNCLASSIFIED

77-03

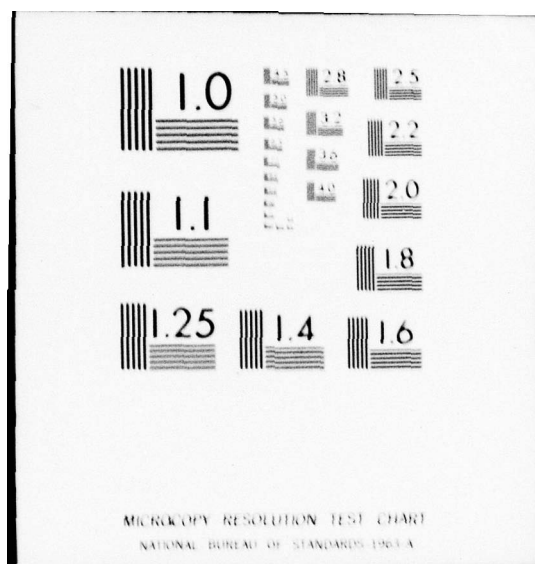
ECOM-75-0650-F-2

DAAA25-75-C-0650

NL

1 of 4
AD
A056 660







LEVEL

12
SR

AD A 056660

Research and Development Technical Report
ECOM - 75-0650-F-2

AUTOMATIC TEST DESIGN



Cihan Tinaztepe

MOORE SCHOOL OF ELECTRICAL ENGINEERING
Department of Computer & Information Science
University of Pennsylvania
Philadelphia, Pa

June 1978

Final Report for Period 30 Jun 75 - 31 Aug 77

PREPARED FOR

ECOM

DISTRIBUTION STATEMENT

Approved for public release:
distribution unlimited.

US ARMY ELECTRONICS COMMAND FORT MONMOUTH, NEW JERSEY 07703

78 07 25 006

AD No. _____
DDC FILE COPY

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The citation of trade names and names of manufacturers in this report is not to be construed as official Government indorsement or approval of commercial products or services referenced herein.

Disposition

Destroy this report when it is no longer needed. Do not return it to the originator.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| 18 19 REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM | | | | | | | | | | | | | | | |
|--|---|--|--------------------|-------------------|--------------------|----------------|------------------------|-----------------|--------------------------------|-----------------|-----------------|-------------------|---------|-----------------|----------------------|-------------------|----------------------|
| 1. REPORT NUMBER ECOM 75-0650-F-2 | 2. GOVT ACCESSION NO. | 3. RECIPIENT'S CATALOG NUMBER | | | | | | | | | | | | | | | |
| 4. TITLE (and Subtitle) AUTOMATED TEST DESIGN | 5. TYPE OF REPORT & PERIOD COVERED FINAL Report 30 Jun 75 - 31 Aug 77 | 6. PERFORMING ORG. REPORT NUMBER 14 17-03 | | | | | | | | | | | | | | | |
| 7. AUTHOR(s) Cihan/Tinaztepe | 8. CONTRACT OR GRANT NUMBER(s) DAAA-25-75-C-0650 | | | | | | | | | | | | | | | | |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS Moore School of Electrical Engineering Department of Computer and Information Science University of Pennsylvania, Philadelphia, PA | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 6.63.6.2A 1G6 63622 AJ29 00 001 | | | | | | | | | | | | | | | | |
| 11. CONTROLLING OFFICE NAME AND ADDRESS US Army Electronics Command ATTN: DRSEL-TL-MS Fort Monmouth, NJ 07703 | 12. REPORT DATE June 78 | 13. NUMBER OF PAGES 294 10 343 p. | | | | | | | | | | | | | | | |
| 14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) | 15. SECURITY CLASS. (of this report) UNCLASSIFIED | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE | | | | | | | | | | | | | | | |
| 16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited. | | | | | | | | | | | | | | | | | |
| 17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report) 16 1G663622AJ29 17 PP | | | | | | | | | | | | | | | | | |
| 18. SUPPLEMENTARY NOTES The work covered by this report was sponsored by Frankford Arsenal, US Army, Philadelphia, PA. However, since the Frankford Arsenal mission relative to this work was transferred to the US Army Electronics Command (ECOM), Ft. Monmouth, NJ, this report is being issued as an ECOM report. | | | | | | | | | | | | | | | | | |
| 19. KEY WORDS (Continue on reverse side if necessary and identify by block number) <table border="0"> <tr> <td>Ambiguity Analysis</td> <td>Component Failure</td> <td>Failure Definition</td> </tr> <tr> <td>Analog Testing</td> <td>Computer-Aided Network</td> <td>Failure Symptom</td> </tr> <tr> <td>Automatic Test Equipment (ATE)</td> <td>Analysis (CANA)</td> <td>Fault Diagnosis</td> </tr> <tr> <td>Automatic Testing</td> <td>Entropy</td> <td>Fault Isolation</td> </tr> <tr> <td>Catastrophic Failure</td> <td>Equivalence Class</td> <td>Fault Isolation Tree</td> </tr> </table> | | | Ambiguity Analysis | Component Failure | Failure Definition | Analog Testing | Computer-Aided Network | Failure Symptom | Automatic Test Equipment (ATE) | Analysis (CANA) | Fault Diagnosis | Automatic Testing | Entropy | Fault Isolation | Catastrophic Failure | Equivalence Class | Fault Isolation Tree |
| Ambiguity Analysis | Component Failure | Failure Definition | | | | | | | | | | | | | | | |
| Analog Testing | Computer-Aided Network | Failure Symptom | | | | | | | | | | | | | | | |
| Automatic Test Equipment (ATE) | Analysis (CANA) | Fault Diagnosis | | | | | | | | | | | | | | | |
| Automatic Testing | Entropy | Fault Isolation | | | | | | | | | | | | | | | |
| Catastrophic Failure | Equivalence Class | Fault Isolation Tree | | | | | | | | | | | | | | | |
| 20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A methodology for designing tests to diagnose and isolate failures in analog circuits has been developed and implemented as a computer program. The input to the program consists of a circuit description and a list of possible failures as changes to the nominal unit under test. The output of the program consists of a report showing how well the failures can be isolated, the tests to be performed, and the diagnosis selection logic which combines these tests to isolate the faulty components. The final output is produced in the (continued) | | | | | | | | | | | | | | | | | |

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

19. KEY WORDS (continued)

Nonprocedural Test Specification

Nonlinear Analysis Program (NAP2)

Nonprocedural OPAL (NOPAL)

Operational Performance Analysis Language (OPAL)

Test Design

Test Module

Test Specification Generation

Test Strategy

Unit Under Test (UUT)

Worst-Case Analysis

20. ABSTRACT (continued)

NOPAL (nonprocedural OPAL) test specification language.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

CONTENTS

| | | |
|-----------|--|-----|
| CHAPTER 1 | STATEMENT OF PROBLEM | |
| 1.1 | Objectives and Motivation | 1 |
| 1.2 | Contributions | 1 |
| 1.3 | Organization | 5 |
| CHAPTER 2 | STATE-OF-THE-ART | 9 |
| 2.1 | Survey Papers and Reports | 11 |
| 2.2 | Analog Fault Isolation Methods | 12 |
| 2.3 | Analog Test Programming Languages | 13 |
| 2.4 | Handbooks | 15 |
| 2.5 | ATE Hardware | 17 |
| CHAPTER 3 | OVERVIEW OF AUTOMATIC TEST PROGRAMMING | 19 |
| 3.1 | Automatic Testing Environment | 20 |
| 3.2 | Automated Test Design and Programming | 24 |
| 3.3 | The NOPAL Language | 32 |
| 3.4 | Development History | 35 |
| 3.5 | Advantages and Disadvantages of the FITS Methodology | 36 |
| 3.5.1 | Advantages | 36 |
| 3.5.2 | Disadvantages | 38 |
| 3.6 | Computer System Requirements | 39 |
| CHAPTER 4 | OVERVIEW OF THE FITS SYSTEM | 40 |
| 4.1 | System Initialization | 43 |
| 4.1.1 | Inputs Required of the User | 43 |
| 4.1.1.1 | Circuit Description | 43 |
| 4.1.1.2 | Available Test Terminals | 50 |
| 4.1.1.3 | Failure Definitions | 52 |
| 4.1.1.4 | Fault Isolation Objectives | 55 |
| 4.1.1.5 | Measurement Accuracy | 58 |
| 4.1.1.6 | Initial Conditions | 61 |
| 4.1.2 | Input File Initialization | 63 |
| 4.1.3 | Generation of Failure Dictionary | 63 |
| 4.1.4 | Failure Dictionary | 64 |
| 4.1.5 | Generation of Candidate Tests | 73 |
| 4.1.6 | NAP2 Candidate Tests Library | 77 |
| 4.1.7 | NOPAL Candidate Tests Library | 79 |
| 4.2 | Failure Simulation and Symptom Generation | 79 |
| 4.2.1 | Computer Aided Network Analysis | 79 |
| 4.2.2 | Generation of the Failure Symptom Table | 82 |
| 4.2.3 | Failure Symptom Table | 82 |
| 4.3 | Decision Limit Determination and Ambiguity Analysis | 84 |
| 4.3.1 | Decision Limit Finder | 84 |
| 4.3.2 | Assertion Limit and Binary Valued Decoder Table | 90 |
| 4.3.3 | Test Limit and Multiple Valued Decoder Table | 91 |
| 4.3.4 | Ambiguity Analysis | 93 |
| 4.3.5 | Ambiguity Report | 94 |
| 4.3.6 | Ambiguity Summary | 99 |
| 4.4 | Optimization to Minimize Test Length | 101 |
| 4.4.1 | Minimization of the Number of Test Setups | 106 |
| 4.4.2 | Minimization of the Number of Assertions | 109 |
| 4.4.3 | Minimization of the Number of Tests per Diagnosis | 111 |

| | |
|---------------------------------|---|
| ACCESSION for | |
| NTIS | White Section <input checked="" type="checkbox"/> |
| DDC | Buff Section <input type="checkbox"/> |
| UNANNOUNCED | <input type="checkbox"/> |
| JUSTIFICATION | |
| BY | |
| DISTRIBUTION/AVAILABILITY NOTES | |
| Date | |

| | | |
|-----------------|--|-----|
| 4.5 | NOPAL Output Generation | 114 |
| 4.5.1 | Output Processors | 114 |
| 4.5.2 | NOPAL Table Output | 116 |
| 4.5.3 | NOPAL Specification Output | 119 |
| CHAPTER 5 | DESIGN OF THE FITS SYSTEM | 133 |
| 5.1 | Types of Circuits to be Tested | 134 |
| 5.2 | Types of Failures | 137 |
| 5.3 | Test Setup | 142 |
| 5.4 | Assumptions | 144 |
| 5.5 | Measurements to Detect Symptoms of Failure | 145 |
| 5.5.1 | Test Limits Specification | 148 |
| 5.5.2 | Calculation of Parameter Limits | 151 |
| 5.6 | Abstraction of Fault Isolation Logic | 155 |
| 5.7 | A Method for Selecting Efficient Set of Tests | 161 |
| CHAPTER 6 | PROGRAM DESCRIPTION | 164 |
| 6.1 | System Initialization | 165 |
| 6.2 | Failure Simulation and Symptom Generation | 174 |
| 6.3 | Decision Limits Determination and Ambiguity Analysis | 180 |
| 6.4 | Optimization to Minimize Test Length | 187 |
| 6.5 | NOPAL Output Generation | 192 |
| CHAPTER 7 | CONCLUSION AND SUGGESTIONS FOR FUTURE RESEARCH | 196 |
| 7.1 | Conclusion | 196 |
| 7.2 | Suggestions for Future Research | 199 |
| ACKNOWLEDGEMENT | | 201 |
| REFERENCES | | 202 |
| APPENDIX A | USER'S GUIDE TO FITS | 211 |
| 1.1 | Preparation of Input for FITS | 211 |
| 1.1.1 | Circuit Description Section | 213 |
| 1.1.1.1 | Overview of NAP2 Input | 213 |
| 1.1.1.1.1 | General Description of NAP2 | 214 |
| 1.1.1.1.2 | Input Organization | 214 |
| 1.1.1.1.3 | Circuit Elements | 215 |
| 1.1.1.1.4 | Function Definition Capability | 219 |
| 1.1.1.1.5 | Built-in Semiconductor Models | 219 |
| 1.1.1.1.6 | Outputs | 226 |
| 1.1.1.1.7 | Initial Conditions | 227 |
| 1.1.1.1.8 | Run Controls | 227 |
| 1.1.1.1.9 | Types of Analysis | 228 |
| 1.1.1.1.10 | Application Example | 231 |
| 1.1.1.2 | Circuit Description Input Section | 254 |
| 1.1.2 | Test Terminals Input Section | 258 |
| 1.1.3 | Failure Definition Additions Input Section | 259 |
| 1.1.4 | Fault Isolation Objectives Input Section | 269 |
| 1.1.5 | Accuracy Specifications Input Section | 271 |
| 1.1.6 | Initial Conditions Input Section | 273 |
| 1.2 | Job Setup Configurations | 280 |

| | | |
|------------|--|-----|
| APPENDIX B | EBNF REPRESENTATION OF NOPAL | 284 |
| APPENDIX C | FITS SYSTEM TAPE CONTENTS | 294 |
| FIGURES: | | |
| CHAPTER 1 | None | |
| CHAPTER 2 | None | |
| CHAPTER 3 | | |
| 3.1 | Essential Constituents of ATE | 23 |
| 3.2 | Phases in Test Program Generation | 25 |
| 3.3 | Top Level Structure of NOPAL Language in EBNF | 34 |
| CHAPTER 4 | | |
| 4.1 | Top Level Flowchart of FITS | 42 |
| 4.2 | UUT-CPS Circuit Diagram | 47 |
| 4.3 | NAP2 Circuit Description of UUT-CPS | 48 |
| 4.4 | UUT-CPS Test Terminals Input | 51 |
| 4.5 | Graphical Representation of Objectives | 57 |
| 4.6 | UUT-CPS Objectives Input | 58 |
| 4.7 | UUT-CPS Accuracy Input | 62 |
| 4.8 | UUT-CPS Initial Conditions Input | 62 |
| 4.9 | Pictorial Representation of Candidate Tests and Test Strategies in FITS | 74 |
| 4.10 | NAP2 Candidate Tests Library Organization | 78 |
| 4.11 | NOPAL Candidate Tests Library | 80 |
| 4.12 | Pictorial Representation of Regions | 85 |
| 4.13 | Fault Isolation Tree | 102 |
| 4.14 | Entropy Evaluation Output | 107 |
| 4.15 | Fault Isolation Tree for the UUT-CPS Example | 108 |
| 4.16 | NOPAL Specification Output | 125 |
| CHAPTER 5 | | |
| 5.1 | Test Setup | 143 |
| 5.2 | Component Tolerance Distributions | 147 |
| 5.3 | Parameter Tolerance Limits | 149 |
| 5.4 | Decision Limit Selection | 150 |
| 5.5 | Symptom Space in 2-D | 159 |
| 5.6 | Address Space | 160 |
| CHAPTER 6 | | |
| 6.1 | System Initialization | 166 |
| 6.2 | Failure Simulation and Symptom Generation | 175 |
| 6.3 | NAP2 Program Structure | 177 |
| 6.4 | Decision Limit Determination and Ambiguity Analysis | 181 |
| 6.5 | Optimization to Minimize Test Length | 188 |
| 6.6 | NOPAL Output Generation | 193 |
| CHAPTER 7 | None | |
| APPENDIX A | | |
| A.1 | General Organization of the Input File | 212 |
| A.2 | Reference Directions | 218 |
| A.3 | PN-Junction Diode Models | 221 |
| A.4 | Bipolar Junction Transistor Models | 223 |
| A.5 | Junction Field Effect Transistor Models | 225 |
| A.6 | Class-B Amplifier Circuit Diagram | 232 |

| | | |
|-------|---|-----|
| A.7 | Test Circuit for Transistor Parameters | |
| A.7a | Test Circuit for IC vs. VCE Characteristics and h_e -parameters | 238 |
| A.7b | Test Circuit for Capacitance Characteristics | 238 |
| A.8 | Input to NAP2 to Analyze the Circuit Shown in Figure A.7 | 239 |
| A.9 | NAP2 Output from the Input Shown in Figure A.8 | |
| A.9a | IC vs. VCE Characteristic | 240 |
| A.9b | h_e -parameter vs. IC Characteristic | 241 |
| A.9c | Capacitance Characteristic | 242 |
| A.10 | Input to NAP2 to Analyze the Circuit Shown in Figure A.6 | 246 |
| A.11 | NAP2 Output from the Input Shown in Figure A.10 | |
| A.11a | Optimal Design of the Operating Point | 247 |
| A.11b | Sensitivity and Worst-Case Analysis | 248 |
| A.11c | Temperature Variation | 249 |
| A.11d | Fourier Analysis of Output Voltage V12 | 250 |
| A.11e | Time Domain Analysis of Sine Input | 251 |
| A.11f | Class-B Operation of the Amplifier | 252 |
| A.11g | Time Domain Analysis of Pulse Input | 253 |
| A.12 | Simple Run Configuration | 281 |
| A.13 | Alternate Configurations to Generate NOPAL Specifications | 283 |

TABLES:

| | | |
|------------|---|-----|
| CHAPTER 1 | None | |
| CHAPTER 2 | None | |
| CHAPTER 3 | None | |
| CHAPTER 4 | | |
| 4.1 | Catastrophic Failure Definitions in FITS | 49 |
| 4.2 | Options in the Failure Dictionary | 54 |
| 4.3 | Options in Accuracy Specifications | 60 |
| 4.4 | Failure Dictionary of the UUT-CPS | 66 |
| 4.5 | Failure Symptom Table | 83 |
| 4.6 | Assertion Limit and Binary Valued Decoder Table | 88 |
| 4.7 | Test Limit and Multiple Valued Decoder Table | 92 |
| 4.8 | Ambiguity Report | 95 |
| 4.9 | Ambiguity Summary | 100 |
| 4.10 | NOPAL Tabular Specification | 118 |
| CHAPTER 5 | None | |
| CHAPTER 6 | | |
| 6.1 | Data Fields in Failure Dictionary | 169 |
| 6.2 | Test Strategy Control Options | 172 |
| 6.3 | Data Fields in the Failure Symptom Table | 180 |
| 6.4 | Data Fields in Assertion Limit and Binary Valued Decoder Table | 184 |
| 6.5 | Data Fields in the Test Limit and Multiple Valued Decoder Table | 185 |
| 6.6 | Data Fields in the Diagnosis Matrix | 192 |
| CHAPTER 7 | None | |
| APPENDIX A | | |
| A.1 | Diode Parameters | 222 |
| A.2 | Bipolar Junction Transistor Parameters | 224 |

| | | |
|------|---|-----|
| A.3 | Junction Field Effect Transistor Parameters | 226 |
| A.4 | Catastrophic Failure Definitions in FITS | 257 |
| A.5 | Options in the Failure Dictionary | 265 |
| A.5a | Component Name Options | 266 |
| A.5b | Failure Function Options | 266 |
| A.5c | Node Options | 267 |
| A.5d | Change Options | 267 |
| A.5e | Type Options | 268 |
| A.5f | Parameter Options | 268 |
| A.6 | Options in Accuracy Specifications | 272 |

CHAPTER 1

STATEMENT OF PROBLEM

1.1 OBJECTIVES AND MOTIVATION

The objectives of the research and development undertaken in this work lie within the scope of total automation of software development for automatic test equipment. This report is concerned with the concept of automation of test design for diagnosing and isolating faulty circuit components or subcircuits in electronic analog circuits.

The design of diagnostic and fault isolation tests is widely recognized as a conceptually difficult but integral part of electronic equipment design. The complex design task is further compounded by the need to use computer controlled automatic test equipment which performs the testing repetitively in production or maintenance environments.

The introduction of a computer to the testing process necessitates the preparation of a new computer program for each different unit under test (UUT). Even slight modifications in the circuit design or in the failures of a UUT may require the generation of a totally different test program.

The automation of the design and programming of tests for digital circuits has received considerable attention. There are several commercially available automatic test program generation systems for digital circuits. CAPS-VIII by

GenRad, TESTAID-III by Hewlett-Packard, D-LASAR by Digitest, CC-TEGAS3 by Control Data Cybernet Service and FLASH by Micro Systems are some of the well known systems. Perhaps the success and wide acceptance of digital test equipment is primarily due to the availability of automatic test program generators. Much less effort has been spent on analog fault diagnosis and isolation, and even less on hybrid circuit (circuits consisting of analog and digital subcircuits).

There are many techniques proposed for detecting failures in analog circuits [1]. Very few of them have actually been programmed for use with automatic test systems. A company which reportedly developed such a system was not willing to provide more information on the methodology they used. There is no reference to the success or failure of this system [2]. Only two computer aided network analysis programs have the built-in capability failure analysis (ASTAP by IBM and UCCAP by University Computing Company). These circuit analysis programs have very limited failure analysis capability. They are primarily directed toward the problem of circuit design rather than testing.

A large number of digital circuit design and test programs were developed by ATE manufacturers for the purpose of producing test programs economically and making ATE more attractive to potential users. However, none of the circuit analysis programs were developed by ATE manufacturers. This explains why computer-aided design has not found direct application in the generation of analog test programs. For

a long time it has been accepted that analog circuit test programs could generally be prepared in a time and cost that the market could accept. Recently this justification has become obsolete because of rising labor costs. It is estimated that 80% of all maintenance costs consist of labor expenses while only 20% are due to replacement parts [3]. The U.S. Defense Department spends annually on maintenance and support an excess of one third of the amount spent on procurement of electronic equipment [4]. According to the U.S. Navy one way of reducing this cost disproportion is to simplify design, introduce modular concepts, minimize software complexity, and increase online testing with the use of automatic test equipment [4]. Consequently, automatic testing and the programming of techniques for analyzing and isolating component, subsystem, and system faults quickly and with minimal human intervention have emerged as a means of reducing costs.

Several recent papers acknowledge the fact that, to date, there are no automatic test program generation systems for analog circuits [2, 4, 5, 6]. A survey conducted by IEEE in 1975 among 5000 automatic test system users indicated that more than one half of them expected new developments in automatic test program generation and fault diagnosis and isolation programs for the future ATE [5]. About one fourth of the users anticipated that advances in automatic test program generation systems will have the greatest impact on future ATEs. Some of the other results of this

survey of interest to this work are as follows: (1) All of the users expressed an interest in the standardization of automatic test programming languages. (2) All of the users felt that the needs of testing complex devices and systems are not satisfied by the present ATE technology. (3) Only about 5% of the users anticipated progress in computer-aided design and signal analysis. The last statistic alone indicates that the advances made in computer-aided network analysis has reached saturation. The other statistics are indicative of the need and the interest in automatic test program generation systems.

The methods currently employed in analog test design are ad hoc and unsatisfactory. For instance, studies during the course of this research indicated that 25% of the catastrophic failures of components could not be detected by manufacturer specified functional tests [7]. Also the methods used to date for preparing ATE programs are considerably behind general computer software technology in other computer application areas. This is in contrast with significant advances being made in ATE hardware area. The developments in hardware are observed as major reductions in cost and in improved transportability.

It is evident that the design and programming tasks represent a major impediment in the critical path to employing automatic test equipment. The need of reducing human activity in preparing and executing test programs to check the performance of devices is well recognized and

requires no further justification.

1.2 CONTRIBUTIONS

The contributions of the methodology developed are two-fold. The first contribution is in the area of electrical engineering. It enables the circuit design engineer to evaluate the circuit design, understand the symptoms of failure, and generate test specifications automatically. It introduces the concept of nonprocedural test specification into test design. Shannon's entropy measure of information content is extended to be a figure of merit in selecting a minimal number of tests. A simple tabular form is developed to specify the test objectives in fault isolation to grade the quality of a test program.

The methodology and its computer implementation (FITS) is a novel approach to automatic test programming. An extensive literature survey indicates that this is the first completed system which takes the circuit description of an analog circuit as input and produces its test specifications as output automatically.

The second contribution is in computer science and relates to automatic programming. When the top and bottom parts of the NOPAL system are viewed together, they exhibit an elegant example of an automatic program generating software system. The minimal input required of the user is accepted in a problem oriented specification language. The

output of the system is a computer program which can be executed in conjunction with automatic test systems. There is no programming knowledge and experience required from its users.

Some more of the application oriented contributions and the capabilities of the methodology developed are summarized below:

1. The system is adaptive to the proficiency of the user. The man-machine interaction is provided through a simple circuit description language. The internal results and outputs of the system are always presented in simple tabular forms to enhance comprehension and enable restart procedures. A proficient user can provide more information and guide the operation of the system.
2. Designing tests with this system does not require special training. The system can be used by a technician having minimal training in electronics. The technician need not have any computer programming experience nor need understand the internal operation of the system.
3. The likelihood of human design error in testing is lessened.
4. Tests designed by test engineers tend to use special interfaces and additional components; on the other hand, the tests produced by this system do not use special

interfaces unless they are included as parts of user defined test strategies.

5. The system selects and uses the most effective test terminals automatically. If the user has no idea of which circuit nodes to make available as external test terminals, then the system may be defaulted to select a minimal number of different test terminals while minimizing the number of different test setup configurations.
6. The failures of a UUT may involve topological or value changes in the circuit description. Cascaded or multiple failures can be diagnosed and isolated if they can be defined as changes to the nominal circuit description.
7. Component and test device tolerances are taken into consideration.
8. All numerical analyses are performed before the test specifications are produced. No time consuming computation has to be done during actual testing. The test program generated can be used in a production environment where the test time duration is an important consideration.
9. The influence of environmental conditions such as temperature, thermal noise, and radiation may be included in the failure analysis.

10. If the user was not able to define some of the possible failures or made errors in modelling the UUT, then some of these conditions can be detected during testing by special provisions in the test specifications.
11. As new failures of a UUT are discovered or eliminated, new test programs can be easily generated.
12. Since the test program is generated automatically, there are no syntactical and programming logic errors.
13. The tests generated by the system are modular, incremental, and nonprocedural. Modularity means that each test specification is independent of others and can be usefully executed by itself. Incrementality means that new test specifications can be added, or some of the existing ones can be deleted from the specified set. The net effect is to increase or to decrease the level of fault isolation capability, respectively. Nonproceduralness means that these tests may be executed in any sequence with no effect on fault isolation logic. The test execution sequence is a function of component protection, interactiveness or efficiency, and it is not dependent on fault isolation logic.

1.3 Organization

This report is organized in seven chapters. Chapter 1 contains the statement of the problem, objectives, and the motivation for the research. It concludes with the contributions this research has brought to electrical engineering and computer science.

In Chapter 2 the state-of-the-art is reviewed with an extensive list of references. It contains a synopsis of the survey papers and reports, fault isolation techniques, software and hardware related to automatic testing systems.

In Chapter 3 an overview of the automatic test program generation concept is given in view of the NOPAL system. It is concluded with a discussion of the advantages, disadvantages and the computer system requirements of the FITS methodology and its program.

In Chapter 4 the operation of the FITS system is described with the inputs to the programs, the processes that take place, and the outputs produced. This chapter also includes an example of automatic test specification for an analog circuit.

In Chapter 5 some of the basic problems associated with preparing test programs are brought to the attention of the reader. The decisions made in this implementation are also given.

Chapter 6 gives the top level description of the programs found in the FITS system. The programs are described in

sufficient detail to familiarize the reader with the procedures so that the FORTRAN implementation can be read comfortably.

The last chapter (Chapter 7) concludes the report by pointing out the significant aspects of the system. A software system of this magnitude can never be considered complete. Section 7.2 of this chapter proposes ideas for future research.

Appendix A contains the user's guide to the FITS system. Appendix B gives the extended BNF representation of the NOPAL language. Appendix C is the table of contents of the system tape. There is a computer tape which is available separately from the project. This tape contains job control statements, source and object codes for the top part of the NOPAL system for the IBM/370 computers.

CHAPTER 2

STATE-OF-THE-ART

There is a vast amount of literature which relates to the fault isolation techniques used in electronics. This chapter contains a synopsis of some of the relevant publications. Digital fault diagnosis and isolation techniques are not included here because this work is restricted to analog circuits only. Four books on fault detection and diagnosis in digital circuits [8-11] and several IEEE Transactions on Computers special issues on fault-tolerant computing [12, 13] discuss the topic in depth and provide a large number of references.

The state-of-the-art in automatic analog testing is surveyed in two parts. The first part relates to fault isolation techniques and ATE software (Sections 2.1 through 2.4). The second part relates to ATE hardware (Section 2.5). The survey papers and reports are given in Section 2.1. Fault diagnosis and isolation techniques are reviewed in Section 2.2. It is followed by a list of the better known ATE programming languages in Section 2.3. Finally, the handbooks related to the topic are referenced in Section 2.4. Section 2.5 contains a short survey of ATE hardware.

2.1 Survey Papers and Reports

"Survey of Current Fault Isolation Techniques" [14] contains a brief but complete synopsis of the pioneering work done in the field until 1962. "The Formulation of Automatic Checkout Techniques" [15] reviews several fault isolation techniques with emphasis on the techniques for interpreting the test results and optimization of test procedures rather than the derivation of tests.

In a NASA report titled "Fault Isolation Computer Methods" (1972), 67 papers are surveyed and classified under ten methods [16]. This classification is followed in Section 2.2. This report is a uniform presentation of some of the well known fault isolation techniques. Each method is presented as follows. First, a general description of the method is given. It is followed by a computer application example and discussion of the results. The types of circuits and the failures which may be detected are also described. Finally the computer requirements are given.

A collection of papers relating to hardware, software, and management aspects of automatic test equipment was edited by Liguori and published by IEEE Press in 1974 [17]. It represents a broad survey of the state of art in automatic test systems.

Several articles published in IEEE Spectrum also survey ATE systems. An article by Eleccion published in August 1974 [18] reviews the economical implications of using ATE

and describes some of the military ATE hardware. A second article published in the next issue is tutorial in nature [19]. The June 1976 issue contains the results of a survey taken among 5000 ATE users [20].

"Circuits Manufacturing" - a monthly magazine of electronics production, assembly and test - contains articles about ATE in about every issue [21]. The magazine emphasizes digital test systems; nevertheless, it is a good source to follow the recent developments in ATE products. The June 1974 issue contains a survey of over 50 automatic test systems, and the January 1977 issue contains a long list of test equipment manufacturers.

2.2 Analog Fault Isolation Methods

1. The classification method is based on recognizing regions of measurements of normal or malfunctioning operation of the system [22]. The FITS methodology can be classified under this method.
2. The key element search method finds the most likely faulty component by determining a smallest index for each component where the index is a function of the measurements and of only one component failure [23].
3. The iterative method is based on the idea that the topology of the circuit and the component values are known at some time before testing, but the failure

functions (component value changes only) are unknown during testing [24, 25]. The most likely failure is found by performing computer simulation while changing the component values to match measurements obtained during testing.

4. The transfer function method is divided into two techniques. The first technique uses the Bode diagram to obtain failure symptoms in the frequency domain. The difference between gains is used to identify the faulty component [26, 27]. The second technique uses the concept of tracking the transfer function of the system. The coefficients of the transfer function are used in a mathematical relationship to determine the faulty component [28, 29].
5. The scalar remnant method defines a scalar performance indicator which is positive when the coefficients of the differential equation describing the system are within prescribed tolerances; otherwise it is negative, indicating a malfunction [30].
6. The inverse probability method uses the difference between actual measurements and the nominal measurement to calculate a probability measure to indicate the most likely failure [31].
7. The power spectra method is based on finding the transfer function through the calculation of the power

spectra to determine if the system is operating nominally [32,33].

8. The parameter identification method identifies the coefficients of the transfer function of the system. The output of the UUT is related to the output of the orthogonal filters which are excited by the same stimulus as the circuit [34]. These coefficients are compared to find the failures.
9. The maximum current method is based on finding the majority current path of the circuit to identify the faulty components along this path [35].
10. The dynamic system testing method uses measurements of the transient response of the system to determine if the response is within acceptable limits [36].

2.3 Analog Test Programming Languages

One of the earlier test programming languages, PLACE (Programming Language for Automatic Checkout Equipment), was developed by the U.S. Air Force to be used with AN/GJQ-9, AN/APQ, AN/GSM-133 and AN/GSM-204(v) test equipment [37].

A volunteer committee organized mainly through ARINC (Aeronautical Radio Inc.) proposed ATLAS (Abbreviated Test Language for All Systems - ARINC Specification 416-5) [38]. ATLAS is a standard abbreviated language for

writing test procedures which can be implemented using either manual or automatic test equipment. The language is intended for defining test requirements with no references to, or dependence on the test equipment used. ATLAS has found wide acceptance among airlines [39, 40], aerospace companies, instrument manufacturers and the military. It is suggested as an industry standard. Both the U.S. Navy and U.S. Air Force have adopted ATLAS as their interim standard for ATE.

On the other hand, the U.S. Army has proposed OPAL (Operational Performance Analysis Language) [41]. It has been predicted that soon the better features of OPAL and ATLAS will be merged to define Department of Defense standard test language [20]. The Automatic Program Generation Project at the University of Pennsylvania has proposed NOPAL as a nonprocedural test specification language [42, 43].

RCA has developed a compiler called UTEC (Universal Test Equipment Compiler) [44]. It is a general purpose compiler which accepts ATE and source language descriptions and produces, as output, an object program from the source language test program.

At Hewlett-Packard, test programs are written in modified ATLAS which is compiled into an intermediate language called ATS-BASIC to be used with their test systems [45].

2.4 Handbooks

"Program Design Handbook For Automatic Test Equipment" by RCA Aerospace Systems is an excellent instruction book about writing test programs. No specific ATE or programming languages are endorsed. It is a result of many years of experience with test programming [46].

There are two significant handbooks which summarize the electronic failure rate data. Useful information on failure rates can be found in "Reliability Stress and Failure Rate Data" MIL-HDBK 217B [47]. Another useful source is the "Failure Rate Data Book" FARADA which includes information on the total number of tests, number of failures, source of data and the environment [48].

"Probabilistic Reliability - An Engineering Approach" by Shooman is a standard reference in reliability analysis [49]. It also includes a short discussion of the failures found in analog circuits.

2.5 ATE Hardware

A recent buyer's guide published in Circuits Manufacturing (January 1977 [50]) lists over 200 manufacturers producing general and special purpose test equipment. A large number of automatic test equipment and their capabilities are listed in Circuits Manufacturing June

1974 [51]. Some of the larger ATE are listed below.

SCATE MARK-VI by General Dynamics is a large general purpose automatic test system [52]. It was originally developed to test the F-111. It is now being marketed for testing analog, digital, navigational and communication systems. ATLAS is the standard programming language.

HP9500 series automatic test systems by Hewlett-Packard is by far the largest and most flexible system on the market today [53]. It is a general purpose test system for analog, digital and mechanical devices. HP ATS-BASIC is the standard programming language.

Another large analog and digital test system is the EQUATE NAFI-ATE supplied by RCA [54]. ATLAS is the standard programming language.

CHAPTER 3

OVERVIEW OF AUTOMATIC TEST PROGRAMMING

This chapter gives an overview of the operations and the methodology employed in a computer aided test design system (FITS) which generates tests to diagnose and isolate failures in analog circuits. These tests are intended to be incorporated into a computer program which will perform fault diagnosis and isolation with computer controlled automatic test equipment.

Section 3.1 presents a brief description of the automatic testing concepts. In Section 3.2 automatic test program generation is described in the context of the NOPAL system. Section 3.3 contains a short description of the NOPAL language. Section 3.4 presents the development history of the NOPAL system. In Section 3.5 some of the advantages and disadvantages of the FITS system are discussed. Finally, in Section 3.6 the computer system requirements to execute the FITS system on different computer installations are described.

3.1 Automatic Testing Environment

In the context of this work, a test means a process to be performed to obtain information about the performance of a component or device. A component may be a simple two terminal artifact such as a resistor or diode. A device may be a circuit board, a module of an electronic assembly, or a complete unit such as a radio set. Whether it is a basic electronic component, or a complex device which is to be tested, it is called a unit under test (UUT).

There are essentially two purposes of testing: (1) Functional testing has the purpose of determining whether the UUT is operational or not (fault diagnosis). The UUT is then classified as good or bad. (2) Testing for fault isolation has the purpose of aiding in repair. In this case testing is directed towards identifying the faulty component in a bad UUT (fault isolation).

Testing may be performed manually or automatically. In manual testing, a test technician uses individual pieces of test equipment such as power supplies, signal generators, oscilloscope, voltmeter, ohmmeter, and a collection of clip leads to connect the test devices to the UUT. The technician may be provided with a test protocol which contains instructions on how to use the test equipment and how to evaluate the results obtained.

In automatic testing, a test procedure is performed with the aid of a computer. The UUT is connected to the test equipment through a switching system, and interface if

necessary. The test protocol a test technician uses is replaced by a computer test program.

The essential constituents of automatic test equipment (ATE) are shown in Figure 3.1. Automatic testing starts after connecting the UUT to the test equipment through an interface. UUT-ATE interface provides the means of connecting the UUT to the test equipment should their connecting points (terminals) be not compatible. After the UUT is properly identified, testing starts. Under the control of the test program, the computer sends instructions to the switching unit to provide the routing of the signals between the test devices and the UUT by setting up connections between their terminals (test setup). There are two types of test devices: (1) stimulus devices are those devices which generate stimuli, such as power supplies, waveform generators, and external loads, and (2) measurement devices are those devices which quantify the response of a UUT to stimuli. The response of a UUT may be a directly measurable physical quantity such as voltage, or it may be a derived quantity from physical measurements such as resistance, complex-impedence, power-gain, or phase-shift.

In automatic testing, control signals from a digital computer can automatically set the programmable test devices to the desired functions with the proper scale. After a stimulus device is instructed to provide a stimulus, the requested stimulus is sent through the established route in the switching unit to the UUT. A command to the measurement

device causes a measurement to be taken at the selected UUT terminal. In most cases, the measurement obtained can be represented by a single number. The term measurement is used to mean either the physical test setup situation, or the number indicating the value of the measurement taken. In context, the usage of this term is unambiguous. The measurement taken is then transferred to the computer and compared to predetermined test limits. If the measurement is within the specified limits for that test, then the instructions in the "go" path are performed. If the measurement obtained is beyond the measurement limits, then the "no-go" path is taken. Following the instructions of the test outcome, the ATE either rearranges to a new test setup, or sends a message to the ATE operator. A new setup may involve a rearrangement in the switching system, and changing of the stimulus and measurement devices. A message to the operator may contain diagnostic information or it may request intervention. Automatic testing proceeds according to the next instruction of the test program.

It is the primary concern of this report to show how test specifications to diagnose and isolate failures in analog circuits are designed automatically. Another report describes how these test specifications are put into an efficient execution sequence and written in the OPAL test programming language [55].

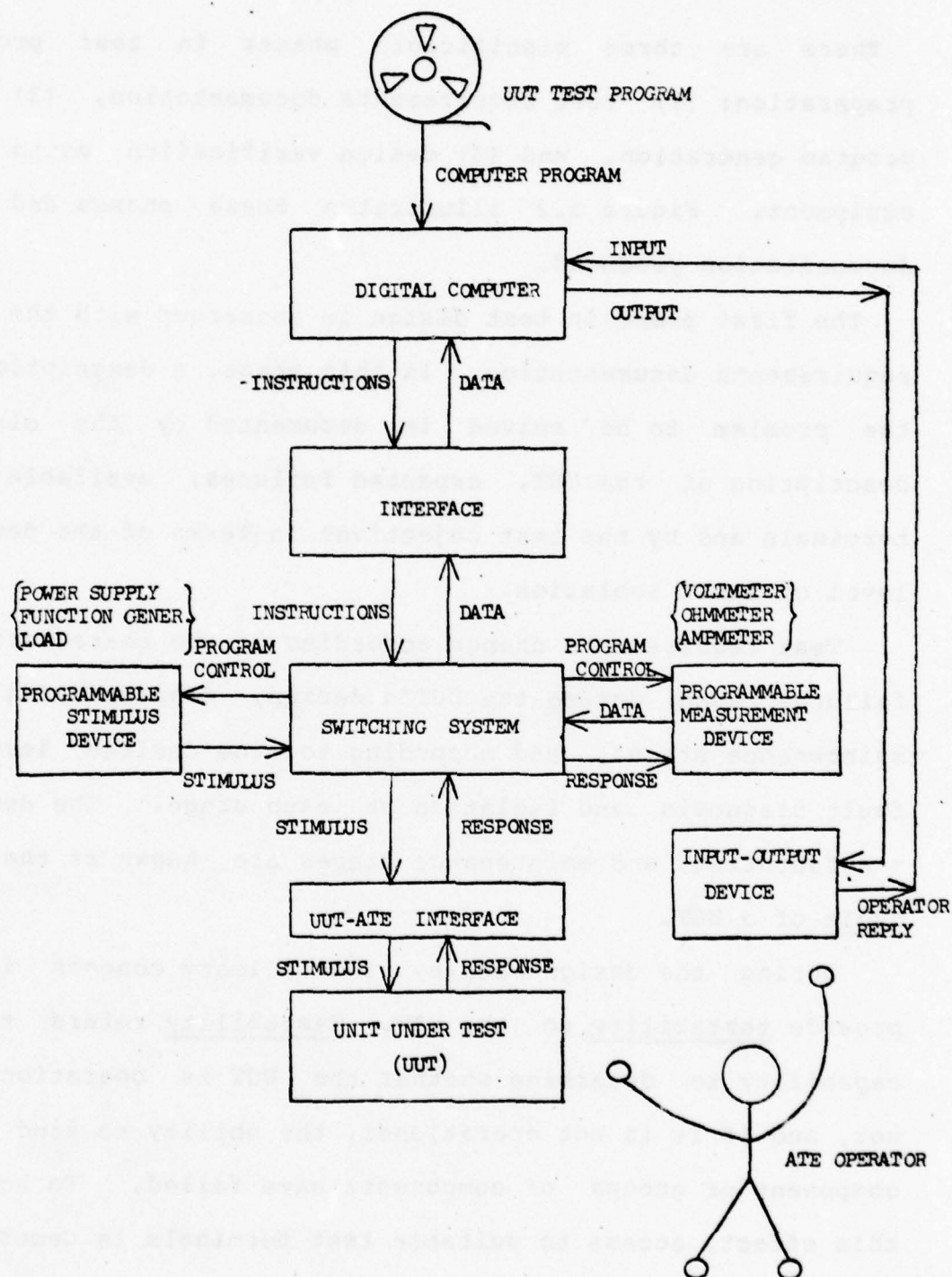


Figure 3.1 Essential Constituents of ATE

3.2 Automated Test Design and Programming

There are three significant phases in test program preparation: (1) test requirements documentation, (2) test program generation, and (3) design verification using test equipment. Figure 3.2 illustrates these phases and the documentation produced.

The first phase in test design is concerned with the test requirements documentation. In this phase, a description of the problem to be solved is documented by the circuit description of the UUT, expected failures, available test terminals and by the test objectives in terms of the desired level of fault isolation.

Test requirements change according to the characteristic failures found during the UUT's design, manufacturing, and maintenance stages, and according to the desired level of fault diagnosis and isolation at each stage. The design, manufacturing, and maintenance stages are known as the life cycle of a UUT.

During the design stage, the primary concern is to provide testability to the UUT. Testability refers to the capability to determine whether the UUT is operational or not, and if it is not operational, the ability to find which component or groups of components have failed. To achieve this effect, access to suitable test terminals is necessary.

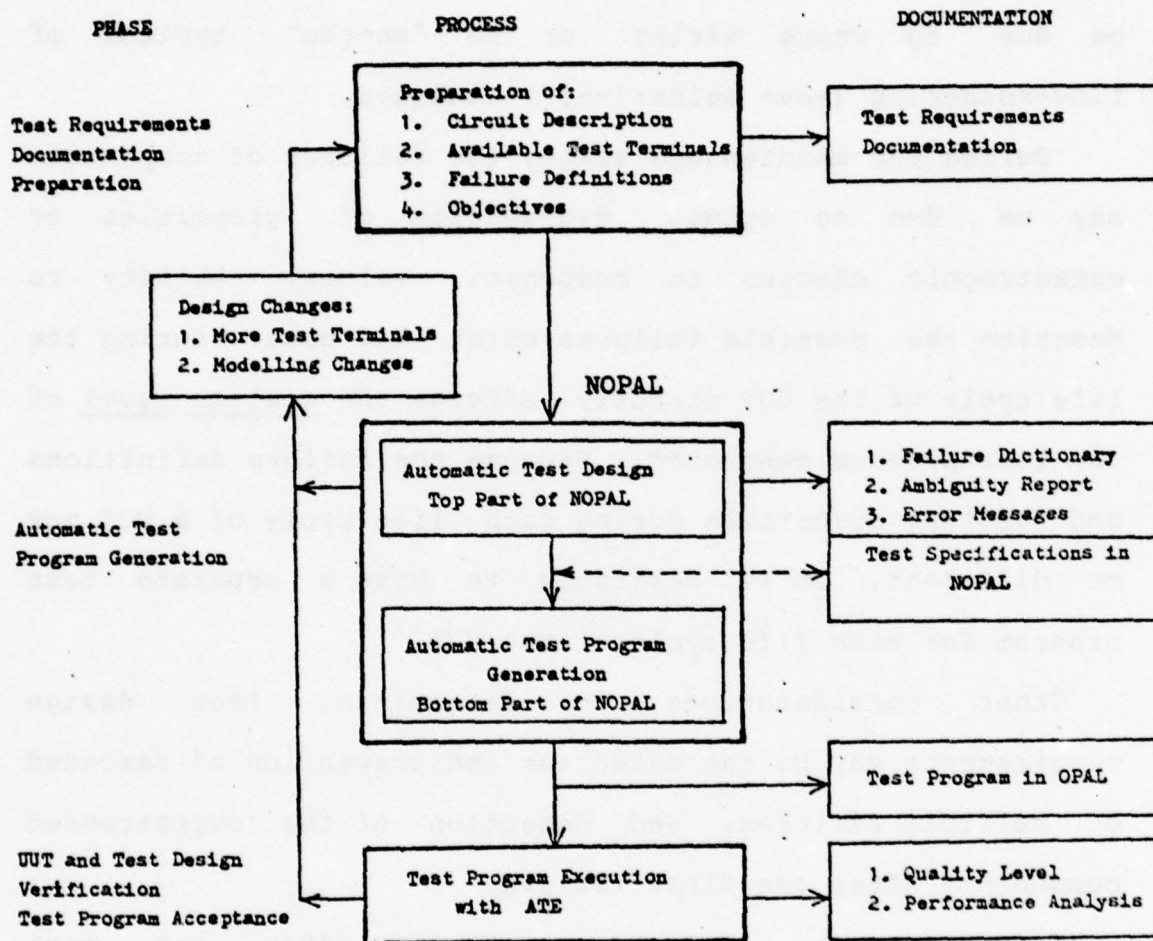


Figure 3.2 Phases in Test Program Generation

During the manufacturing stage, in addition to the typical component failures which may exist in the incoming component lots from manufacturers, predictable production line failures may be present in the UUT. These failures may be due to wrong wiring or to "shorts" typical of flow-soldering (wave soldering) techniques.

During the maintenance stage, the failures of components may be due to aging, degradation of properties or catastrophic changes in component values. Ability to describe the possible failures which may occur during the life cycle of the UUT directly affects the quality level of the test program generated. Because the failure definitions and the test objectives during each life cycle of a UUT may be different, it is desirable to have a separate test program for each life cycle of the UUT.

Other considerations in determining test design requirements may be the detection and prevention of cascaded or multiple failures, and detection of the overstressed components after the first failure.

Test program generation starts after the test requirements are determined. This phase is shown in the central box in Figure 3.2. Based on the UUT design information, the NOPAL (nonprocedural OPAL) automatic test program generation system produces an object program in OPAL (Operational Performance Analysis Language) [55]. The main advantage of OPAL over earlier test programming languages (such as ATLAS [56]) is the facility for modular

development of test programs. The test modules can be developed and shared by different programmers.

The NOPAL system consists of top and bottom parts. Each one of these parts is independent of the other and could be usefully employed by itself. The top part is used to design test specifications to diagnose and isolate the failures of a prospective UUT. The bottom part accepts the test specifications generated by the top part as input and puts them into an efficient execution sequence. The output is a test program in OPAL. Tests can also be prepared manually with NOPAL.

The interface between the top and the bottom parts is provided by a number of test specifications written in the NOPAL language. Unlike OPAL, NOPAL is not a programming language (A summary of the NOPAL language is given in Section 3.3). It is a method for describing individual tests and diagnoses. It is nonprocedural in the sense that it does not have facilities for controlling the execution sequence of the tests. The concept of NOPAL is central to the methodology that has been developed, and therefore the entire system is referred to as the NOPAL system. Test specifications in NOPAL consist of test modules. Each test module is independent of other test modules. A test module describes a single test which has the objective of diagnosing some failure modes.

The most important properties of these test specifications are nonproceduralness, incrementality, and

modularity. Nonproceduralness means that these tests may be executed in any sequence. The fault isolation logic is independent of the execution sequence. Incrementality means test specifications may be added or deleted without changing previously existing test modules. Modularity refers to the fact that each test specification is self contained.

When the top part of NOPAL is used independently of the bottom part, it is abbreviated FITS (Functional and Fault Isolation Test Specification). This report is concerned only with the top part of NOPAL. The bottom part is described elsewhere [57,58,59].

In the top part of NOPAL, test design is accomplished partly manually and partly automatically. Circuit description, test objectives indicating the desired level of fault diagnosis and isolation, and possible failures of the UUT need to be determined and prepared manually as input to the FITS system. However, majority of the inputs are assigned default values using only the circuit description of the UUT and "common" failures of the basic circuit components. Thus, the user has to indicate only the additional failures and the desired fault isolation level if they differ from the standard (system default) assignments.

A test design has two significant stages. First, the response of the UUT with failure to stimuli is simulated using a computer aided network analysis (CANA) program to generate a data base containing symptoms of failures. Then in the second stage, similar failure symptoms are combined

together to obtain test limits and diagnosis information. From these results, a collection of test specifications and associated diagnoses are generated. If these test specifications and diagnoses cannot satisfy the test objectives, changes in the circuit design should be made and a new test design should be initiated.

The top part of NOPAL provides detailed reports documenting the failures under consideration, the fault isolation level and the test specifications generated by the system to meet the test objectives.

The bottom part of NOPAL functions as an automatic program generator. The inputs are test specifications written in NOPAL. The execution sequence of tests is determined automatically for maximum efficiency as summarized below. The output is a complete ATE test program in the procedural language OPAL.

The first component of the bottom part performs syntactical analysis of the test specifications. Syntactical errors and documentation consisting of a specification listing and cross reference reports formatted for easy readability are produced. The second component incorporates an engineering knowledge-base needed to optimize the execution sequence of the test modules. In the course of the analysis, the system produces various additional reports including errors and warnings of detected inconsistencies, fault isolation summary, and precedence analysis showing the test execution sequence. When the

input to the bottom part is prepared automatically by the top part, no syntactical or logic errors are expected. The last component of the bottom part generates a test program in OPAL which is acceptable to ATE. The program is compiled by an OPAL compiler, and then it is ready to test UUTs repeatedly.

The six sequencing strategies in the bottom part are summarized here for completeness. (1) Data determinancy incorporates the principle that data must be generated before it can be used. The generation of data by a predecessor test module is recognized by the declaration of a target variable. A successor test module must then have the same variable declared as a source variable. (2) Interactiveness is dictated by the need to exchange messages interactively with the ATE operator. (3) Component protection is based on the concept that non-destructive testing can be achieved if a critical component is tested before other components which depend on it for their normal operation. (4) Fault isolation strategy schedules testing in a top-down fashion using component subset relationships. The more general fault isolation tests are executed first. Then follow the lower level, more specific tests, on the condition that a failure is detected on the top level. (5) Stimulus application is concerned with the efficient application of stimuli. It is based on the assumption that the application of stimuli is the most time consuming process; and hence, it is preferable to conduct all the

possible tests, once a stimulus is applied. (6) Failure likelihood uses the idea that efficiency is obtained by first testing the components which are more likely to fail. Using these strategies, the execution sequence of tests are optimized. The reader can find the complete description of the bottom part of NOPAL in another report [59].

The third and last phase in test program generation relates to the verification of the test program design. Before any test program can be used with confidence, its quality level in actual operation needs to be determined. This is done by checking the program performance with representative UUTs having no failures and with UUTs having known failures to exercise all parts of the test program. Failure diagnosis and isolation statistics derived from a sufficient number of trials are used to evaluate the test design. Its quality level is determined by comparing the performance to the acceptable quality level (AQL) specifications. The sampling plan to be used in this phase may be dictated as a part of the test design requirements, or one of the well established standard procedures outlined in MIL-STD-105 [60] may be employed.

Only after the test program has exhibited acceptable performance can it be used to check out the UUTs coming from production lines or for repair and maintenance purposes.

3.3 The NOPAL Language

The overall structure of the NOPAL language is summarized in Figure 3.3 using extended BNF notation. The complete description can be found in the NOPAL user's guide [58]. As shown in the top line, a specification in NOPAL has three parts: test, UUT, and ATE. The latter two are intended to provide UUT and ATE independence, in the sense that changes that occur in a UUT or an ATE can be reflected only in these parts of the specification, and other parts may remain unaltered.

The concept of test module, as an independent integral entity with the corresponding diagnoses which identify failures, is central to the NOPAL system. Each test module is independent of other test modules. Individual test modules can be modified, deleted, or added without affecting other test modules. A test module describes a single test which has the objective of diagnosing some failure modes.

A test module consists of four parts: stimuli, measurements, logic, and diagnoses. Stimuli and measurements consist of two types of expressions named conjunction and assertion. A conjunction is a list of waveforms associated with stimuli or measurement devices and respective test points of the UUT where they are applied. Assertions are expressions composed of variable names and logical and arithmetic operators. The assertions are used for two purposes. The primary use is for stating the ranges of measurements which determine the passing or failing of a

test. The other use is to evaluate dynamically, at the test time, the parameters of respective stimulus or measurement devices.

The logic part determines the selection of diagnoses based on the outcome of one or more tests. Finally, the diagnosis part identifies the failures and the messages which communicate the results of testing to the operator of an ATE.

```

<NOPAL SPECIFICATION> ::= <TEST SPECIFICATION>
                           <UUT SPECIFICATION>
                           <ATE SPECIFICATION>

<TEST SPECIFICATION> ::= <TEST MODULE> [<TEST MODULE>]*

<TEST MODULE>          ::= [<STIMULI>] [<MEASUREMENT>]
                           [<LOGIC>]* [<DIAGNOSIS>]*

<STIMULI>              ::= [<CONJUNCTION>] [<ASSERTION>]*

<MEASUREMENT>          ::= [<CONJUNCTION>] [<ASSERTION>]*

<CONJUNCTION>          ::= <TEST POINTS> <RELATION> <WAVEFORM>
                           [<TEST POINTS> <RELATION> <WAVEFORM>]*

<LOGIC>                ::= <OPERATOR> <DIAGNOSIS ID>

<DIAGNOSIS>            ::= <DIAGNOSIS ID> <MESSAGE>
                           [<FAILURE IDS>]
                           [<OTHER DATA>] [<TIMING>]
                           [<OPERATOR RESPONSE>]

```

```

KEY:  [ ... ]  OPTIONAL
      [ ... ]* MAY REPEAT ZERO OR MORE TIMES

```

FIGURE 3.3 Top Level Structure of NOPAL Language in EBNF

3.4 Development History

The research and development reported in this work has been conducted by the Automatic Program Generation Project at the Department of Computer and Information Science, Moore School of Electrical Engineering, University of Pennsylvania. This phase of the research was started in late January 1976 and was completed (Version 1.0) in March 1977.

FITS - the top part of NOPAL, is written entirely in standard FORTRAN and runs on UNIVAC 90 and IBM 370 computers. The bottom part of NOPAL, the Automatic Test Program Generation System, has been implemented in PL/I. Both systems were developed as parts of doctoral research. The test programs generated by the NOPAL system are intended to operate commercially available automatic test equipment such as General Dynamic's SCATE-MARK VI [61] and Hewlett-Packard's HP9500 [62].

The FITS system has been tested with several examples. One example consisted of a power supply control circuit made of 10 semiconductors and 15 other components with a total of 60 failures. It took about 90 minutes of IBM S/360 Mod 65 time (CPU and Channel) to generate satisfactory test specifications.

3.5 Advantages and Disadvantages of FITS Methodology

Some advantages and disadvantages of the methodology are discussed in this section.

3.5.1 Advantages

Advantages of the methodology developed for test design in this work are listed below:

1. Designing tests with the FITS system does not require specially skilled test engineers. In most cases a circuit description of the UUT may already be available in a CANA language if the circuit design engineer has used computer simulation to check the design.
2. Tests designed by test engineers generally utilize special interfaces and external load to the UUT. On the other hand, the tests specified by the FITS system do not use special interfaces unless they are necessitated as requirements of user defined test strategies.
3. Likelihood of human test design error is reduced.
4. FITS aids the test designer in selecting more effective test terminals.
5. Ambiguity in fault isolation and failure modes which cannot be diagnosed are known at any time during the test design process.
6. Tests generated by the FITS system are modular, incremental and nonprocedural. Modularity means that each test is independent of others and defines a

complete test process by itself. Incrementality means that new tests may be added to or deleted from the set of tests specified. The net result of of this operation is only an increase or decrease in the level of fault isolation. Nonproceduralness means that tests may be executed in any sequence with no effect on the fault isolation logic.

7. Although the system was initially developed to isolate single catastrophic failures of individual network components, other failure modes such as out-of-tolerance (degraded) components or multiple and cascaded failures may be defined and isolated by the FITS system.
8. Component tolerance, stimulus and measurement device inaccuracies are taken into consideration.
9. Since the test specifications are generated automatically, there are no syntax and logic errors in the NOPAL test specifications produced.
10. All numerical analyses are performed before the tests are generated; therefore, no time consuming computation has to be done during actual testing. This implies that the test design is suitable to be employed in a production environment.
11. Environmental conditions such as ambient temperature, radiation effects, and thermal noise effects may be included in the simulation.
12. Some of the failure modes which are unknown to the FITS system can be detected. When this situation is

encountered an appropriate message is produced.

3.5.2 Disadvantages

The disadvantages of the FITS system are:

1. It requires accurate modelling of the semiconductor devices.
2. A large number of computer simulations are needed.
3. All of the UUT failure modes must be known and defined before test design. This is, however, a drawback of all failure isolation methods reported in the literature [63].
4. In an actual testing environment, the UUT may have failure modes which are not previously declared to the FITS system. In this case, there are two possible outcomes; (1) The logic in the test specifications will recognize the unknown failure mode as known and classify it into one of the failure classes, or (2) the specifications will not be able to classify it as belonging to one of the known failure classes and will give a message indicating this. It is impossible to predict which of the two outcomes will take place before the undefined failure is analyzed.

3.6 Computer System Requirements

The FITS system was developed on a UNIVAC 90/70 and trial runs were made on an IBM S/360 Mod 65. Both of these computers are medium size computer systems. All programs are written in ANSI FORTRAN. In total, they are about 15000 statements long.

Most of the programs in FITS require less than 100Kbytes of memory. The NAP2 circuit analysis program requires about 100Kbytes of memory. Actually the program is longer than this, but an efficient overlay structure makes it possible to run within 100Kbytes. NAP2 is being modified to run on General Automation GA18/30 and DEC PDP11 minicomputers in less than 32Kbytes of memory [64]. FITS was developed with the realization of this fact, so that it can be converted to run on minicomputer systems as well.

The FITS system creates about 10 small internal files which are stored as sequential disk files. The output of simulation may sometimes be very large. Hence it is recommended that NAP2 output be saved on a computer tape, and made a permanent file for later reference.

CHAPTER 4

OVERVIEW OF THE FITS SYSTEM

In this chapter, a user view of the FITS system is presented in terms of the inputs required, the processes that take place and the outputs produced to document the test design. The failure symptom generation methodology is based on simulating the UUT response to stimuli and measurements derived from three test strategies. The fault isolation methodology is based on finding regions of normal operation and regions of malfunction of the UUT.

The complete design process is divided into five major phases as illustrated in the central column of Figure 4.1. They are : (1) system initialization, (2) failure simulation and symptom table generation, (3) decision limit determination and ambiguity analysis, (4) optimization of the test specifications, and (5) NOPAL test specifications generation.

The inputs to the FITS system are described in Section 4.1. As shown on the left side of the flowchart of FITS (Figure 4.1), there are six inputs to the system. They are: (1) circuit description of the UUT, (2) externally accessible test terminals, (3) definition of failures particular to the UUT, (4) test objectives in terms of the desired level of failure isolation, (5) measurement accuracy in terms of scale accuracy and number of significant digits, and (6) initial conditions of the UUT. Of the six inputs,

only the first one is necessary. The remaining five can be automatically generated either from the circuit description or from other "common practice" engineering knowledge. The user can easily modify these system generated (default) assignments by inserting the desired changes, additions, or deletions to the proper computer files.

The outputs produced by the FITS system are described after the process which generate them. The six significant outputs of FITS are shown on the right side of Figure 4.1. They contain the following information: (1) failure dictionary contains the definitions of the failures, (2) candidate tests libraries contain the tests to be tried, (3) failure symptom table contains characteristic responses of the UUT at the test terminals, (4) decision limits and decoder tables show the measurement ranges used to identify the failures and how to get diagnostic information out of them, (5) ambiguity report shows how the failures are isolated into groups using the test limits specified in the fourth output, and (6) NOPAL specification of tests communicate the tests to be performed to the bottom part of NOPAL.

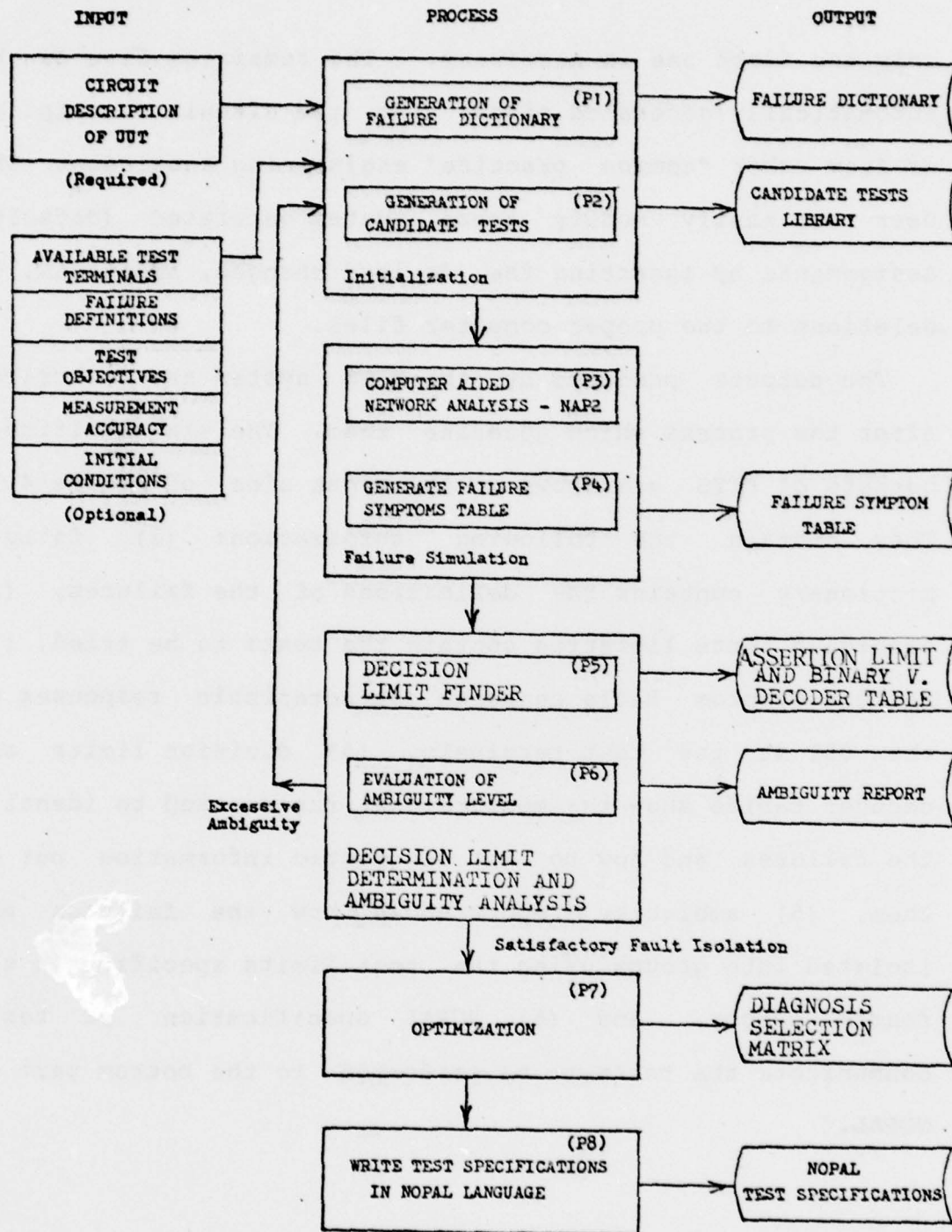


Figure 4.1 Top Level Flowchart of FITS

4.1 System Initialization

The system initialization process relates to the preparation of data and input to the other parts of the FITS system. It has three subprocesses: (1) input initialization, (2) generation of failure dictionary and (3) generation of candidate tests. Each one of these subprocesses are described with the inputs they take and the outputs they produce.

4.1.1 Inputs Required of the User

The six input sections required of the user are described in Sections 4.1.1.1 through 4.1.1.5. The first input is required, and the remaining five are optional. The FITS system assigns default values to any input (except circuit description) when it is not specified.

4.1.1.1 Circuit Description

The behavior of the UUT needs to be simulated under nominal and malfunctioning conditions before test specifications can be determined. For this purpose an accurate circuit description of the UUT must be supplied to the computer aided network analysis (CANA) program within the FITS system.

The first step taken in this process is to prepare sufficiently accurate models of the components used in the circuit diagram. The equivalent model of the circuit diagram may consist of resistors, capacitors, inductors,

mutual inductances, voltage and current sources, bipolar and FET devices. Accurate modelling of the last two components may be an involved task. However, in the case of popular device types, a number of published models are available [65,66]. For example, a library containing the most popular types of transistors and diodes may be found in a model library on the EXTENDED SCEPTRE program [66] tape. Complex devices are modelled by simpler equivalent circuits. The behavior of an equivalent circuit element may be defined by a numerical constant, table, or mathematical expression. Component tolerances need to be specified by stating the maximum expected deviation from the nominal value. After an equivalent circuit has been prepared, each circuit node is given an arbitrary name. A circuit node is defined as the point of common potential at the junction formed by interconnection of two or more circuit elements. Each component is assigned a unique name. The first letter of the component name identifies the component type as well. Current flow directions and source polarities are also indicated. In the UUT circuit description no power supplies and input signal sources are specified. The connection of power supplies and other stimuli including the status of mechanical switches, potentiometers and other adjustable components are treated as different initial conditions of the UUT. The initial conditions are provided as a separate input. Details of circuit description for circuit analysis can be found in the user manual (Appendix A and NAP2 User's

Manual). The inputs to most circuit analysis programs have similar syntax. If the user is familiar with any CANA system, it should be very easy to write circuit descriptions in the language of the circuit analysis program found in FITS. Mainly for reasons of convenience, the NAP2 [67] circuit analysis program is used in this implementation.

In this chapter test specifications to diagnose and isolate the catastrophic failures in a control power supply (CPS) circuit are used to provide examples. The schematic of the CPS circuit is shown in Figure 4.2. This is the power supply control circuitry of the laser range finder AN/VVS-1. Details of its operation can be found in MIL-C-14866(MU) [68]. The NAP2 circuit description of the CPS circuit is shown in Figure 4.3. In this example, simplified semiconductor models are used to make the description shorter. The first line provides identification names to the FITS system. This line indicates that the following input is the circuit description of the unit under test named "UUT_CPS". Its failure dictionary is called "DEFAULT". The remainder of an input line after a colon (:) is interpreted as comment by NAP2. Lines 9 through 17 are the simplified semiconductor models. Lines 19-45 are the descriptions of the components of the circuit. The first component is a resistor called R1 which is incident on the circuit nodes 23 and 7. Its nominal value is 680 ohms. "FAILS" is a keyword recognized by the automatic dictionary generation program. It generates the default catastrophic

failure modes of this type of component, namely open and short. Lines 47-50 show the assignment of tolerance to the circuit components. For example, resistors R2 and R3 both have $\pm 1\%$ tolerance.

The user indicates which circuit components may fail by suffixing the component description in the circuit description input by the "FAILS" command which is optionally followed by "DEFAULT" (or no entry). This means the component may have any one of the catastrophic default failure modes described in Table 4.1. The names of nonstandard failures are entered similarly after the "FAILS" command. This option allows the user to define any modification of the nominal circuit description to be a failure mode of the UUT. The nonstandard failure definitions must be supplied completely by the user in the Failure Definitions input section. The default catastrophic failure definitions are built into the system so that all necessary instructions to NAP2 to modify the nominal circuit description are automatically generated.


```

-----
CIRCUIT DESCRIPTION UUT CPS DEFAULT (FAILURE MODES)
:REFER TO UUT CIRCUIT DIAGRAM CONTROL POWER SUPPLY #10559261
:REPORT NUMBER: MIL-C-14866(MU), 2 MARCH 1970
:A UNIT OF THE LASER RANGE FINDER AN/VVS-1 PROJECT NO 1240-A135
:UUT TERMINAL NAMES <-> CIRCUIT NODE NAME
: (J1_A <-> 1), (J1_B <-> 0), (J1_E <-> 5), (J1_F <-> 29),
: (J1_G <-> 33), (J1_H <-> 26), (J1_X <-> 32)
:SIMPLIFIED SEMICONDUCTOR MODELS - USED FOR ILLUSTRATION ONLY
  Q1132/PNP/  AF .99  AR .39  IS 99E-15  CE 80E-12  CC 4 5E-12
  Q1613/NPN/  AF .99  AR .10  IS 9E-15   CE 5E-12   CC 10E-12
  Q1724/NPN/  AF .98  AR .50  IS 9E-15   CE 275E-12 CC 15E-12
  Q2060/NPN/  AF .99  IS 1E-15          CE 85E-12  CC 15E-12
  D3070/DIODE/ IS 0.1E-15  CJ 5E-12
  D3600/DIODE/ IS 100E-15  CJ 2.5E-12
  D202A/DIODE/ IS 50E-15   CJ 1E-12
  D1202/DIODE/ IS 10E-15   CJ 1E-12
  D4247/DIODE/ IS 1E-15    CJ 1E-12
:RESISTORS & CAPACITOR
  R1  23  7  680 : FAILS
  R2  19  9  301 : FAILS
  R3   9  0 1470 : FAILS
  R4  23 11 1000 : FAILS
  R5  10  0 1000 : FAILS
  R6  23 20  50 : FAILS
  R7  21 25 1000 : FAILS
  R8  29 12  270 : FAILS : SHORT(R8) CANNOT HAPPEN
  R9  14 32  200 : FAILS
  R10 23 17 10000 : FAILS
  R11 17 18 23 700 : FAILS
  RSW 18  0 0.1 : OPEN(RSW) IS SAME AS OPEN(R11) >
               : SHORT(RSW) IS SAME AS NOMINAL
  RLD  1  0 2200 : FAILS
  C1  12  0 10E-6 : FAILS : OPEN(C1) IS SAME AS NOMINAL
:DIODES
  TDCR1  7 26 D3070 : FAILS
  TDCR2  7 19 D3600 : FAILS : OPEN(TDCR2) IS SAME AS OPEN(R2)
  TDCR3 13 14 D3600 : FAILS : OPEN(TDCR3) IS SAME AS OPEN(R9)
  TDCR4 26  1 D202A : FAILS
  TDCR5 32  1 D202A : FAILS
:TRANSISTORS
  TQ1A 11  9 10 Q2060 : FAILS
  TQ1B 23 12 10 Q2060 : FAILS
  TQ2  20 11 21 Q1613 : FAILS : COLL OPEN(TQ2) IS SAME AS OPEN(R6)
  TQ3  32 21 25 Q1724 : FAILS : BE SHORT(TQ3) IS SAME AS OPEN(R7)
  TQ4  13 17 23 Q1132 : FAILS : COLL OPEN(TQ4) IS SAME AS OPEN(R9)
:COMPONENT TOLERANCES
*MODIFY 2 0.01 R2 R3
*MODIFY 3 0.05 R1 R4 R5 R6 R7 R8 R10 R11
*MODIFY 4 0.0467 REV15V.E
*MODIFY 5 0.0345 REV29V.E
:END OF CIRCUIT DESCRIPTION
-----

```

Figure 4.3 NAP2 Circuit Description of UUT-CPS

TABLE 4.1 CATASTROPHIC FAILURE DEFINITIONS IN FITS

| CIRCUIT ELEMENT | FAILURE FUNCTION | DEFINITION |
|-----------------------------|------------------|--|
| Resistor | OPEN | Replace by a high valued resistor |
| | SHORT | Replace by a small valued resistor |
| Capacitor | OPEN | Replace by a high valued resistor |
| | SHORT | Replace by a small valued resistor |
| Inductor | OPEN | Replace by a high valued resistor |
| | SHORT | Replace by a small valued resistor |
| Diode | OPEN | Replace by a high valued resistor |
| | SHORT | Replace by a small valued resistor |
| Bipolar Junction Transistor | COLL_OPEN | Insert a high valued resistor in series with the collector |
| | BASE_OPEN | Insert a high valued resistor in series with the base |
| | EMIT_OPEN | Insert a high valued resistor in series with the emitter |
| | BC_SHORT | Short base collector with a small resistor |
| | BE_SHORT | Short base emitter with a small resistor |
| | EC_SHORT | Short emitter collector with a resistor |

4.1.1.2 Available Test Terminals

A test terminal is defined as a circuit node of a UUT which is externally available for attaching test devices. Not all of the nodes of a UUT may be available for stimulus and measurement device connections. The terminals which are available for test device connection must be stated so. All terminals, where the power supplies are to be connected should be externally accessible since a UUT is expected to contain no internal power supplies. The common terminal (ground) should be declared as a test terminal as well.

When the user does not provide this input, all circuit nodes are assumed to be externally available as test terminals. The system then selects and references only those test terminals which minimize the number of test setups. However, the user must be cautious because this approach will result in a large number of potentially applicable tests which in turn will consume excessive computer time during the simulation process.

Figure 4.4 illustrates the specification of test terminals in the CPS example. The first line indicates that the following input is the description of the test terminals available on the UUT connector named "J1-PIN". In the remaining lines the test terminals are described by the circuit node name followed by the corresponding external UUT connector pin name. The rest of an input line contains comments about the test terminal.

| TEST TERMINALS J1_PIN <CKT NODE><TEST TERM>[<COMMENT>] | | |
|--|------|----------------------------|
| 0 | J1_B | GROUND |
| 1 | J1_A | Q SWITCH |
| 25 | J1_E | +24V AUX. |
| 29 | J1_F | +15V |
| 23 | J1_G | +29V |
| 26 | J1_H | +28,+24 TANK BATTERY SENSE |
| 32 | J1_X | BT1_RECHARGEABLE BATTERY |

Figure 4.4 UUT-CPS Test Terminals Input

4.1.1.3 Failure Definitions

In general, a failure is recognized by three properties: (1) an action or event, (2) a standard against which to compare the event, and (3) a procedure to measure the event against the standard. A failure of a UUT is a failure event by definition. The nominal circuit behavior (i.e., circuit with no failures) is the standard against which the failures are compared to determine their diagnosability. Since failure is only a relative concept, any component value may be declared to be the cause of UUT malfunction. For example, in some applications, the change of the value of a resistor by a small percentage, say 10%, may be a failure. In other applications 10% change of component values may be acceptable. Sometimes failure definitions may involve topological changes in the circuit description. A typical case is the failure of capacitors in power supply circuits. These capacitors generally fail by the puncture of the isolation layers (shorting). The equivalent circuit component is a small value resistance replacing the capacitor.

A failure mode of a UUT is defined as the state a UUT is in when one or more of its components fail in a predetermined fashion. A failure definition of a component is the circuit description of the equivalent circuit component or components capable of representing the behavior of the failed component. A failure mode may be identified descriptively such as "SHORT(R1)". It may also be

identified by the sequence number assigned to it in the failure dictionary. The failure dictionary is a table which lists the failure modes of the UUT by sequence number, descriptive name, and by the definition of the failure in terms of equivalent circuit components. Table 4.2 summarizes the options available in the failure dictionary entries. The organization of this table is described in more detail in Section 4.1.4, and in Appendix A.

To provide consistency between the circuit descriptions language and failure definition, the failures of the UUT are communicated to the FITS system as changes to the nominal circuit description. Hence, a failure mode of a UUT may be viewed as another circuit description.

It is observed that in the majority of cases, the failures of analog circuits tend to be catastrophic (open or short) [69,70,71,72]. In digital circuits similar behavior is observed. To ease the task of failure dictionary preparation, a collection of catastrophic failure definitions are provided by the FITS system (see Table 4.1). The user has to declare only those failures which differ from the default assignments.

After the name of a nonstandard failure of a circuit component is declared in the circuit description input, the description of the failure in NAP2 statements is entered in this section of input. In the CPS example, all failure modes were assumed to be default failures. Therefore, no illustration of this section is given here. Examples to

nonstandard failure definitions can be found in the user's guide in Appendix A.

TABLE 4.2 OPTIONS IN THE FAILURE DICTIONARY

| NAME | DATA TYPE | OPTIONS | DESCRIPTION |
|------------------|--------------------------|-----------------|---|
| ID. | 4 digit integer | not applicable | Failure mode sequence number |
| COMPONENT NAME | string of 12 characters | see own table | Name of the component as it appears in the circuit description |
| FAILURE FUNCTION | string | see own table | Name of the failure function of the failure of the component |
| NODES | 2 or 3 two digit numbers | see own table | Identifies the incidence of component on the circuit nodes |
| CHANGE | string | see own table | Describes the change in the nominal circuit due to the failure |
| TYPE | string | see own table | For simple failures identifies the new component type. Otherwise describes where the definition is to be found. |
| PARAMETER | string or number | see own table | Value of the new component or subcircuit library member name |
| VALUE | string or number | not applicable | Value of the component in the nominal circuit |
| ALIAS | string | string | Alternate failure function name. This name is passed to the bottom part. |
| INDEX | 2 digit integer | 2 digit integer | Failure rate index is not currently supported by FITS. If supplied, it is passed on to the bottom part. |

4.1.1.4 Fault Isolation Objectives (Desired Ambiguity Level)

There are cases when a user is not interested in isolating each failure of the UUT uniquely, but is willing to have them isolated in small groups, mainly for reasons of economy in test program generation time and actual testing time. Unless otherwise requested the FITS system attempts to diagnose better than 85% of the failures declared in the failure dictionary. It uses the same tests for fault isolation. However, the user may modify this requirement by specifying the following statements:

$m\%$ DIAGNOSIS,

$p_1\%$ OF FAILURES IN 1 -AMBIGUOUS CLASSES,

...

$p_k\%$ OF FAILURES IN k -AMBIGUOUS CLASSES,

Diagnosis percentage ($m\%$ diagnosis) means that of all the failures in the failure dictionary, it is sufficient to have only $m\%$ detected as reasons of failure of the UUT. This specification alone gives no restriction on fault isolation. Separate statements convey this information. Fault isolation percentage ($p_i\%$ isolation) is stated in terms of cumulative percentage. Therefore;

$$p_1 < p_2 \dots p_n \text{ and } k_1 < k_2 \dots k_n,$$

where p_i is the cumulative percentage of failure modes which are isolated k_i -ambiguously or better. k -ambiguity is the inability to distinguish between k failure modes. k -ambiguous class of failures is a set of k failure modes

which cannot be distinguished from one another with the given tests. A k-ambiguous class is also called an equivalence class or equivalent class of failures.

The concept of cumulative fault isolation percentage can be graphically represented as shown in Figure 4.5. In this graph, the horizontal axis is labelled by k-ambiguity. The vertical axis shows the cumulative fault isolation percentage. The curve obtained by connecting the points of cumulative percentage versus k-ambiguity is called a fault isolation curve. In order to satisfy the given fault isolation requirement, the FITS system has to find tests whose results yield a fault isolation curve which either matches or exceeds the requested level.

Figure 4.6 shows the objectives specified for the CPS example. The first line indicates that the following input specifies the test objectives called "STANDARD". It requests that the tests designed should be capable of diagnosing 80% of all of the possible failures. 25% of the failures should be uniquely identifiable. Also, 50% should be 2 or less ambiguous, 60% should be 5 or less ambiguous, and 80% should be 10 or less ambiguous.

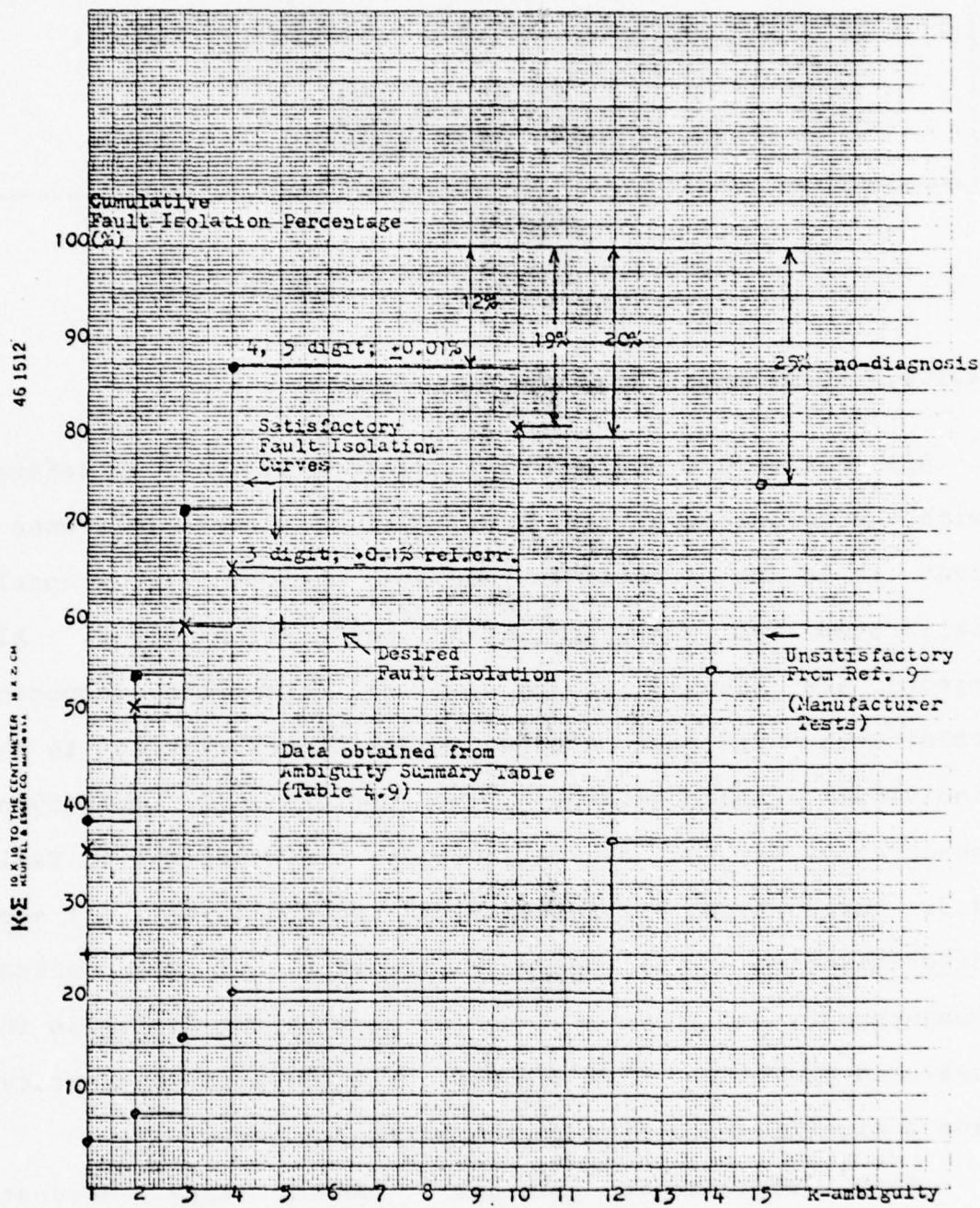


FIGURE 4.5 Graphical Representation of Objectives

| OBJECTIVES | STANDARD | <PERCENT> | <AMBIGUITY> | <COMMENT> |
|------------|----------|-----------|--------------------------|-----------|
| 80.00 | % | DIAGNOSIS | | |
| 25.00 | 1 | 25% | UNIQUELY | |
| 50.00 | 2 | 50% | IN CLASSES OF 2 OR LESS | |
| 60.00 | 5 | 60% | IN CLASSES OF 5 OR LESS | |
| 80.00 | 10 | 80% | IN CLASSES OF 10 OR LESS | |

Figure 4.6 UUT-CPS Objectives Input

4.1.1.5 Measurement Accuracy

All measurement devices have limited accuracy associated with the measurements they make. It is assumed that when a test is to be performed, the test equipment is properly calibrated and makes negligible calibration error. All errors are assumed to be due to noise and component tolerances which have a known range of contribution to the individual measurements. The options in specifying measurement and test design accuracy are described in Table 4.3. Three types of accuracy may be specified: (1) zero discrimination, (2) percent accuracy of the actual measurement, and (3) the number of significant digits in the measured quantity. FITS assumes 0.1% measurement accuracy, and 4 significant digits in the meter readings.

Some measurement devices cannot take accurate measurements near zero; that is, there is a threshold level below which measurements are meaningless. This limit is specified as zero-discrimination. FITS, then, does not use

the responses of the UUT between plus and minus values of the discrimination range. The default value for all measurements is $1.0E-6$.

Figure 4.7 shows the accuracy specification for the CPS example. The first line indicates that the following input is the accuracy specification called "VERY-LOW". The second line indicates that the measurements whose absolute values are less than $1.0E-3$ should not be used in fault isolation. The third line indicates that the measurements obtained by the measurement devices are inaccurate by $\pm 0.1\%$ of the actual measurement. The fourth line indicates that the measurements are accurate to only 3 significant digits.

The remaining lines are program control parameters. It is expected that in most applications the default values will be satisfactory. These parameters are explained in Table 4.3. Their effect may be easier to understand after reading Section 4.3. The first option (SORT) controls the sorting strategy for the assertions created. The second option (OPTIMIZE) allows the user to skip the test length minimization process. Finally, the third option (MISSING) allows the user to equivalence the failures missing from the circuit simulation to a failure which has already been simulated.

TABLE 4.3 OPTIONS IN ACCURACY SPECIFICATIONS

| OPTION NAME | TYPE | DEFAULT | VALUE | PURPOSE |
|------------------------|----------|---------|-------|---|
| ZERO DISCRIMINATION | Flt. No. | 1E-6 | | Measurement whose absolute value is less than this value is not included in creating assertions |
| MEASUREMENT INACCURACY | Flt. No. | 0.1% | | Percent inaccuracy in measurements |
| SIGNIFICANT DIGITS | Int. | 4 | | Number of significant digits in the measurements |
| SORT | Int. | 0 | 0 | List the assertions and tests in the order they are created |
| | | | 1 | Sort the assertions within the tests only according to increasing sensitivity |
| | | | 2 | First sort the assertions within the tests and then sort the tests among themselves |
| | | | 3 | Leave the assertions in the order they were created but sort the tests among themselves |
| | | | 4 | Sort the assertions regardless of the tests |
| OPTIMIZE | Int. | 1 | 0 | Include all tests (or assertions) to select diagnoses |
| | | | 1 | Minimize the number of tests (or assertions) to select diagnoses |
| MISSING | Int. | 1 | <n> | Put the missing failure symptoms in the same region as failure <n> |

4.1.1.6 Initial Conditions

As briefly mentioned earlier the status of mechanical switches, potentiometers, and power supplies define different initial conditions for a UUT. Initial conditions specification serves two purposes. (1) It allows the user to declare the power supplies to be used in the DC-strategy testing (see Section 4.1.5). Using this input, FITS automatically generates the NOPAL and NAP2 statements to try the DC-strategy tests. (2) A modified version of the initial conditions specification allows the user to define tests and initial conditions if the FITS generated standard tests are not sufficient. The user, in this case, has to provide the NOPAL and NAP2 statements which describe the test and the type of analysis. This is a very powerful option which allows the user to try different tests and utilize the results after the fault detection and isolation methodology of FITS.

The details of how to write the initial conditions can be found in Appendix A. Here only an example is presented. Figure 4.8 shows the initial conditions used in the CPS example. The first line indicates that the following input belongs to the initial conditions named "NORMAL". The first initial condition describes a user defined test enclosed between the BEGIN and END statements. The lines which start with a colon in the first column are NOPAL statements. All other lines are interpreted as NAP2

| ACCURACY | VERY-LOW |
|------------|---|
| 1.0E-3 | ZERO DISCRIMINATION |
| 0.1 | MEASUREMENT INACCURACY (PERCENT) |
| 3 | NUMBER OF SIGNIFICANT DIGITS |
| 0 =SORT | LIST ASSERTIONS AND TESTS AS CREATED |
| 1 =OPT | OPTIMIZE ASSERTIONS PER DIAGNOSIS |
| 1 =MISSING | FAILURES (IF ANY) SHOULD NOT BE DIAGNOSABLE |

Figure 4.7 UUT-CPS Accuracy Input

```

INITIAL-CONDITIONS NORMAL
BEGIN DEFINE IMPLIED CROSS IMPEDENCE TEST
: STIMULUS ;
: CONJUNCTION :
: ( < J1_E , J1_G > AMP =
: JSUPPLY ( 1.0E-6 AMP , 1.0E-6 MHO ) );
:
: MEASUREMENT ;
: CONJUNCTION :
: ( < J1_G , J1_B > VOLT =
: VOLTMETER ( VJ1_G1 VOLT ) )
:
: TARGET : VJ1_G1 ;
:
GOHMETER 23 0 1.0E-6 J 1.0E-6
*DC *WORST V23 :IMPLIED CROSS IMPEDENCE
END

BEGIN
.
.
.
END

BEGIN THIS IS A SIMPLE TEST OF POWERING THE UUT
REVB1 32 0 0.01 E 24.0
REV15V 29 0 0.01 E 15.0
REV29V 23 0 0.01 E 29.0
REV24V 26 0 0.01 E 24.0
END
END-INITIAL CONDITIONS

```

Figure 4.8 UUT-CPS Initial Conditions Input
(Partial Listing)

statements of a test.

These initial conditions except the last one are user defined cold circuit tests. The test strategies are described in Section 4.1.5. FITS generates most of the cold circuit tests automatically. In the CPS example, the computer runs were made before the automatic test generator program was implemented. Unfortunately for some tests, terminal connections are reversed. Instead of adapted data, the exact test descriptions are given as user defined tests. The last initial condition specifies the power supplies of the CPS circuit. FITS automatically generates the appropriate NOPAL and NAP2 statements to perform the DC-strategy tests as applicable.

4.1.2 Input File Initialization

This process accepts the user input from a single source file and writes each input section into its own file. Because it has no significance on the methodology, it is not further described.

4.1.3 Generation of Failure Dictionary

The first major operation that takes place when FITS is started is the creation of a failure dictionary which contains all the information relating to the definition of the failure modes of a UUT. The input to this process (P1 in Figure 4.1) is the circuit description of the UUT. The process scans the circuit description and produces a failure

dictionary using the catastrophic failure definitions described in Table 4.1. The user has the option to modify the failure dictionary and to include additional failures.

After the generation of the dictionary is completed, a user oriented version of the failure dictionary table is printed (see Table 4.4).

4.1.4 Failure Dictionary

The failure dictionary is a table generated by the FITS system automatically. It defines the failure modes of a UUT as used by the FITS system. Table 4.4 illustrates the entries in the failure dictionary. The "sequence number" is an integer which is used as a short identifier for the failure mode. The "component name" is the name of the component which has failed as declared in the circuit description. The "failure function" is the descriptive name of the failure of the component. The entries under "nodes" show which circuit nodes this component is incident on. The "change" entry describes the change caused by the failure in the circuit, such as topological or value changes. The "type" entry identifies what the equivalent circuit description is, such as resistor, capacitor, or library model. The "parameter" entry shows the value of the new component or gives the name of the library member which describes a complex failure definition. The "remark" entry is for comments. "Value" entry gives the nominal value or

the model name of the component. "Alias" is an alternate failure function name. "Index" is the failure rate index of the failure mode. In this implementation of FITS all failures are assumed to be equally likely. However, the failure index may be changed to control the bottom part of NOPAL to sequence the execution of the tests such that the tests which isolate the failures with higher failure index are performed before the tests which isolate the failures with the lower failure index. Failure rates of circuit components and mean time between failure (MTBF) are two parameters that are used to calculate failure indices. An introduction to this concept can be found in references [69, 70]. There are a number of references which periodically update component reliability data. Detailed explanation of failure rate equations and other data can be found in MIL-HDBK-217B [74] and in FARADA [75].

Table 4.4 shows the complete failure dictionary for the CPS example. The dictionary was automatically generated from the circuit description given in Figure 4.3. The ordering of the failure modes were slightly rearranged so that the failure sequence numbers match the assignments made in an earlier work [73,76].

TABLE 4.4 FAILURE DICTIONARY OF THE UUT-CPS

FAILURE DICTIONARY --- (PAGE 1)

| ID. | COMPONENT NAME | FAILURE FUNCTION | NODES | CHANGE | TYPE | PARAMETER | (VALUE) | REMARK (ALIAS) | (INDEX) |
|-----|----------------|------------------|-------|--------|----------|-----------|---------|----------------|---------|
| 1 | ALL_COMPS | NONE | | NONE | NONE | NOMINAL | NOMINAL | NOMINAL | 0 |
| 2 | R1 | OPEN | 23 7 | VALUE | RESISTOR | 1.0E9 | 680 | OPEN | 0 |
| 3 | R1 | SHORT | 23 7 | VALUE | RESISTOR | 1.0E-1 | 680 | SHORT | 0 |
| 4 | R2 | OPEN | 19 9 | VALUE | RESISTOR | 1.0E9 | 301 | OPEN | 0 |
| 5 | R2 | SHORT | 19 9 | VALUE | RESISTOR | 1.0E-1 | 301 | SHORT | 0 |
| 6 | R3 | OPEN | 9 0 | VALUE | RESISTOR | 1.0E9 | 1470 | OPEN | 0 |
| 7 | R3 | SHORT | 9 0 | VALUE | RESISTOR | 1.0E-1 | 1470 | SHORT | 0 |
| 8 | R4 | OPEN | 23 11 | VALUE | RESISTOR | 1.0E9 | 1000 | OPEN | 0 |
| 9 | R4 | SHORT | 23 11 | VALUE | RESISTOR | 1.0E-1 | 1000 | SHORT | 0 |
| 10 | R5 | OPEN | 10 0 | VALUE | RESISTOR | 1.0E9 | 1000 | OPEN | 0 |

TABLE 4.4 FAILURE DICTIONARY (Continued)

FAILURE DICTIONARY --- (PAGE 2)

| ID. | COMPONENT NAME | FAILURE FUNCTION | NODES | CHANGE | TYPE | PARAMETER | (VALUE) | REMARK (ALIAS) | (INDEX) |
|-----|----------------|------------------|-------|--------|----------|-----------|---------|----------------|---------|
| 11 | R5 | SHORT | 10 0 | VALUE | RESISTOR | 1.0E-1 | 1000 | SHORT | 0 |
| 12 | R6 | OPEN | 23 20 | VALUE | RESISTOR | 1.0E9 | 50 | OPEN | 0 |
| 13 | R6 | SHORT | 23 20 | VALUE | RESISTOR | 1.0E-1 | 50 | SHORT | 0 |
| 14 | R7 | OPEN | 21 25 | VALUE | RESISTOR | 1.0E9 | 1000 | OPEN | 0 |
| 15 | R7 | SHORT | 21 25 | VALUE | RESISTOR | 1.0E-1 | 1000 | SHORT | 0 |
| 16 | R8 | OPEN | 29 12 | VALUE | RESISTOR | 1.0E9 | 270 | OPEN | 0 |
| 17 | R9 | OPEN | 14 32 | VALUE | RESISTOR | 1.0E9 | 200 | OPEN | 0 |
| 18 | R9 | SHORT | 14 32 | VALUE | RESISTOR | 1.0E-1 | 200 | SHORT | 0 |
| 19 | R10 | OPEN | 23 17 | VALUE | RESISTOR | 1.0E9 | 10000 | OPEN | 0 |
| 20 | R10 | SHORT | 23 17 | VALUE | RESISTOR | 1.0E-1 | 10000 | SHORT | 0 |
| 21 | R11 | OPEN | 17 18 | VALUE | RESISTOR | 1.0E9 | 4700 | OPEN | 0 |

TABLE 4.4 FAILURE DICTIONARY (Continued)

FAILURE DICTIONARY --- (PAGE 3)

| ID. | COMPONENT NAME | FAILURE FUNCTION | NODES | CHANGE | TYPE | PARAMETER | (VALUE) | REMARK (ALIAS) | (INDEX) |
|-----|----------------|------------------|-------|-------------|----------|-----------|---------|----------------|---------|
| 22 | R11 | SHORT | 17 18 | VALUE | RESISTOR | 1.0E-1 | 4700 | SHORT | 0 |
| 23 | RLD | OPEN | 1 0 | VALUE | RESISTOR | 1.0E9 | 2200 | OPEN | 0 |
| 24 | RLD | SHORT | 1 0 | VALUE | RESISTOR | 1.0E-1 | 2200 | SHORT | 0 |
| 25 | C1 | SHORT | 12 0 | TOPOLOGICAL | RESISTOR | 1.0E-1 | 10E-6 | SHORT | 0 |
| 26 | TDCR1 | OPEN | 7 26 | TOPOLOGICAL | RESISTOR | 1.0E9 | D3070 | OPEN | 0 |
| 27 | TDCR1 | SHORT | 7 26 | TOPOLOGICAL | RESISTOR | 1.0E-1 | D3070 | SHORT | 0 |
| 28 | TDCR2 | SHORT | 7 19 | TOPOLOGICAL | RESISTOR | 1.0E-1 | D3600 | SHORT | 0 |
| 29 | TDCR3 | SHORT | 13 14 | TOPOLOGICAL | RESISTOR | 1.0E-1 | D3600 | SHORT | 0 |
| 30 | TDCR4 | OPEN | 26 1 | TOPOLOGICAL | RESISTOR | 1.0E9 | D202A | OPEN | 0 |
| 31 | TDCR4 | SHORT | 26 1 | TOPOLOGICAL | RESISTOR | 1.0E-1 | D202A | SHORT | 0 |
| 32 | TDCR5 | OPEN | 32 1 | TOPOLOGICAL | RESISTOR | 1.0E9 | D202A | OPEN | 0 |

TABLE 4.4 FAILURE DICTIONARY (Continued)

FAILURE DICTIONARY --- (PAGE 4)

| ID. | COMPONENT NAME | FAILURE FUNCTION | NODES | CHANGE | TYPE | PARAMETER | (VALUE) | REMARK (ALIAS) | (INDEX) |
|-----|----------------|------------------|----------|-------------|----------|-----------|---------|----------------|---------|
| 33 | TD6RS | SHORT | 32 1 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q202A | SHORT | 0 |
| 34 | TQ1A | COLL_OPEN | 11 9 10 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q2060 | COLL_OPEN | 0 |
| 35 | TQ1A | BASE_OPEN | 11 9 10 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q2060 | BASE_OPEN | 0 |
| 36 | TQ1A | EMIT_OPEN | 11 9 10 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q2060 | EMIT_OPEN | 0 |
| 37 | TQ1A | BC_SHORT | 11 9 10 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q2060 | BC_SHORT | 0 |
| 38 | TQ1A | BE_SHORT | 11 9 10 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q2060 | BE_SHORT | 0 |
| 39 | TQ1A | EC_SHORT | 11 9 10 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q2060 | EC_SHORT | 0 |
| 40 | TQ1B | COLL_OPEN | 23 12 10 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q2060 | COLL_OPEN | 0 |
| 41 | TQ1B | EMIT_OPEN | 23 12 10 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q2060 | EMIT_OPEN | 0 |
| 42 | TQ1B | BC_SHORT | 23 12 10 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q2060 | BC_SHORT | 0 |
| 43 | TQ1B | BE_SHORT | 23 12 10 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q2060 | BE_SHORT | 0 |

TABLE 4.4 FAILURE DICTIONARY (Continued)

FAILURE DICTIONARY --- (PAGE 5)

| ID. | COMPONENT NAME | FAILURE FUNCTION | NODES | CHANGE | TYPE | PARAMETER | (VALUE) | REMARK (ALIAS) | (INDEX) |
|-----|----------------|------------------|----------|-------------|----------|-----------|---------|----------------|---------|
| 44 | T01B | EC_SHORT | 23 12 10 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q2060 | EC_SHORT | 0 |
| 45 | T02 | BASE_OPEN | 20 11 21 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q1613 | BASE_OPEN | 0 |
| 46 | T02 | EMIT_OPEN | 20 11 21 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q1613 | EMIT_OPEN | 0 |
| 47 | T02 | BC_SHORT | 20 11 21 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q1613 | BC_SHORT | 0 |
| 48 | T02 | BE_SHORT | 20 11 21 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q1613 | BE_SHORT | 0 |
| 49 | T02 | EC_SHORT | 20 11 21 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q1613 | EC_SHORT | 0 |
| 50 | T03 | COLL_OPEN | 32 21 25 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q1724 | COLL_OPEN | 0 |
| 51 | T03 | BASE_OPEN | 32 21 25 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q1724 | BASE_OPEN | 0 |
| 52 | T03 | EMIT_OPEN | 32 21 25 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q1724 | EMIT_OPEN | 0 |
| 53 | T03 | BC_SHORT | 32 21 25 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q1724 | BC_SHORT | 0 |
| 54 | T03 | EC_SHORT | 32 21 25 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q1724 | EC_SHORT | 0 |

TABLE 4.4 FAILURE DICTIONARY (Continued)

FAILURE DICTIONARY --- (PAGE 6)

| ID. | COMPONENT NAME | FAILURE FUNCTION | NODES | CHANGE | TYPE | PARAMETER | (VALUE) | REMARK (ALIAS) | (INDEX) |
|-----|----------------|------------------|----------|-------------|----------|-----------|---------|----------------|---------|
| 55 | TQ4 | BASE_OPEN | 13 17 23 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q1132 | BASE_OPEN | 0 |
| 56 | TQ4 | EMIT_OPEN | 13 17 23 | TOPOLOGICAL | RESISTOR | 1.0E9 | Q1132 | EMIT_OPEN | 0 |
| 57 | TQ4 | BC_SHORT | 13 17 23 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q1132 | BC_SHORT | 0 |
| 58 | TQ4 | EC_SHORT | 13 17 23 | TOPOLOGICAL | RESISTOR | 1.0E-1 | Q1132 | EC_SHORT | 0 |
| 59 | C1 | OPEN | 12 0 | SAMEAS # 1 | RESISTOR | 1.0E9 | 10E-6 | OPEN | 0 |
| 60 | TDCR2 | OPEN | 7 19 | SAMEAS # 4 | RESISTOR | 1.0E9 | D3600 | OPEN | 0 |
| 61 | TDCR3 | OPEN | 13 14 | SAMEAS # 17 | RESISTOR | 1.0E9 | D3600 | OPEN | 0 |
| 62 | TQ2 | COLL_OPEN | 20 11 21 | SAMEAS # 12 | RESISTOR | 1.0E9 | Q1613 | COLL_OPEN | 0 |
| 63 | TQ3 | BE_SHORT | 32 21 25 | SAMEAS # 15 | RESISTOR | 1.0E-1 | Q1724 | BE_SHORT | 0 |
| 64 | TQ4 | COLL_OPEN | 13 17 23 | SAMEAS # 17 | RESISTOR | 1.0E9 | Q1132 | COLL_OPEN | 0 |
| 65 | TQ4 | BE_SHORT | 13 17 23 | SAMEAS # 15 | RESISTOR | 1.0E-1 | Q1132 | BE_SHORT | 0 |

TABLE 4.4 FAILURE DICTIONARY (Continued)

FAILURE DICTIONARY --- (PAGE 7)

| ID. | COMPONENT NAME | FAILURE FUNCTION | NODES | CHANGE | TYPE | PARAMETER | (VALUE) | REMARK (ALIAS) | (INDEX) |
|-----|----------------|------------------|-------|-------------|----------|-----------|---------|----------------|---------|
| 66 | RSW | OPEN | 18 0 | SAMEAS # 21 | RESISTOR | 1.0E9 | 0.1 | OPEN | 0 |
| 67 | RSW | SHORT | 18 0 | SAMEAS # 1 | RESISTOR | 1.0E-1 | 0.1 | SHORT | 0 |

4.1.5 Generation of Candidate Tests

The next process, P2, generates two files. The inputs to the process are the circuit description, the list of externally available test terminals, the failure dictionary, and the initial conditions of the UUT. The first output file contains instructions to a CANA program (NAP2) for simulating a test (Figure 4.11). The second file contains the description of the same stimuli and measurements in the NOPAL language (Figure 4.10).

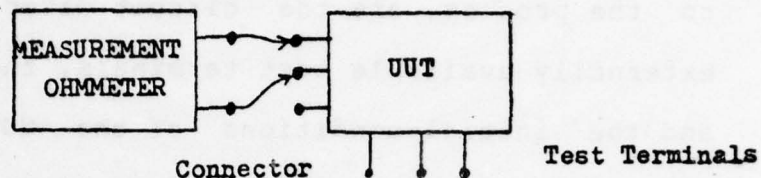
In the present system, there are three test strategies. A test strategy refers to a similar group of tests which apply identical stimulus or measurement devices between UUT's test terminals to determine its response. Users may employ other strategies following generally the methodology outlined below. Such additional strategies may easily be designed and added as new test strategies to the FITS system.

The three strategies are applied in the following sequence: (1) cold circuit (CC), (2) direct current (DC) and (3) user defined (UD) strategies. Pictorial representations of the CC and DC strategies are shown in Figure 4.9.

In the cold circuit strategy of testing, candidate tests are generated which will measure the cold circuit (i.e., all power supplies disconnected) impedance of the UUT across its test terminals.

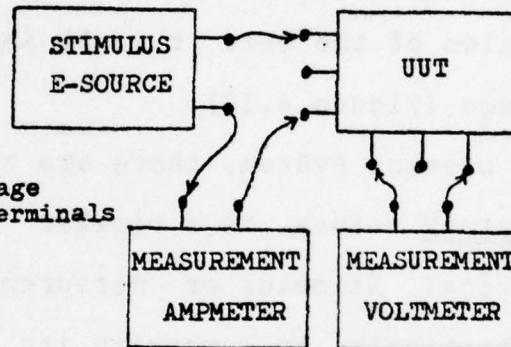
Cold Circuit Strategy:

Measures impedance between the test terminals of the UUT.

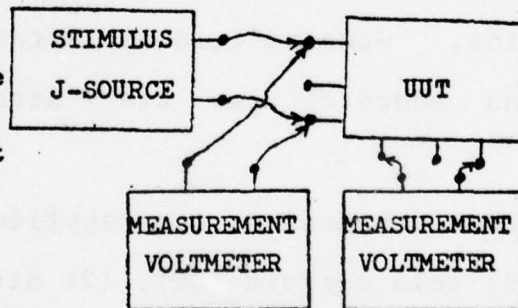


Direct Current Strategy:

If the power supply is voltage source, measure the current through it and also measure the voltage level at all other test terminals



If the power supply is a current source, measure the voltage across it and also measure the voltage at all other test terminals.



User Defined Strategy: Any test which can be realized by the circuit analysis program NAP2 with the *WORST case analysis.

Figure 4.9 Pictorial Representation of Candidate Tests and the Test Strategies in FITS

In the direct current strategy, the power supplies of the UUT are connected, and then, the voltage levels at the test terminals and current through power supplies are measured. Finally, in the user defined test strategy the tests, as defined by the user in the initial conditions input, are evaluated. The user defined tests follow generally the methodology used in the CC and DC strategies. The only restriction on the user defined test strategy is that the measurement made on a UUT should be representable by a range of measurements. Thus, the user can perform frequency (AC), and transient (TR) analysis tests on the UUT. These type of tests are inherently difficult to automate because of the large number of possibilities and usefulness.

The tests to be simulated are saved in the candidate tests libraries as described in Sections 4.1.6 and 4.1.7. Whenever a measurement is found by the CANA program, the result is expressed in terms of minimum and maximum worst-case range of values (see Section 5.5). If the UUT has several initial conditions such as different positions of mechanical switches, or variable resistances, each one is considered to be a distinct initial state of the UUT, and the above strategies are repeated for each initial state.

The NAP2 candidate tests library file contains instructions to the circuit analysis program NAP2 to simulate the circuit response to a desired test (stimulus and measurement) when the UUT has one of the failures defined in the dictionary. The meaning of a test in this section is no

longer a physical arrangement of the ATE, but a description of the stimulus and measurement devices as simplified equivalent circuit components or functions. For example, even though a measurement device such as a voltmeter is a complex device, it is usually described as an idealized voltmeter which has no loading effect on the circuit. The measurement taken by such a meter is found by the circuit analysis program as the voltage difference across two nodes of the circuit.

When a test is to be simulated by the NAP2 program the following information is supplied as input (see Figure 4.9):

1. Nominal circuit description;
2. Definition of a failure which changes the nominal circuit description to make it represent the UUT with a known failure mode;
3. Description of the stimulus and measurement devices and their connections to the circuit;
4. Type of analysis to be performed by the CANA program to find the symptoms of failure. In this implementation only DC worst-case analysis is performed;
5. A "RUN" command to the program to simulate the circuit described.

The range of values a measurement can take is calculated with the circuit analysis program by performing worst-case analysis. Worst case measurements are specified with minimum and maximum limits. These limits are obtained after perturbing all the nominal values of the circuit

components to their tolerance limits (if they have any specified) depending on the sign of the sensitivity of the measurement to the component value. These concepts are described further in Section 5.5.

The data obtained from the simulation of the above input to the CANA program gives the symptom of a single failure mode. It is, therefore, repeated for all failure modes in order to be able to completely specify a test in the failure symptoms table (Table 4.5).

4.1.6 NAP2 Candidate Tests Library

NAP2 candidate tests library is a file produced by the candidate tests generator program. It contains all the necessary instructions for the circuit analysis program to simulate a test and measure the response of the UUT to stimuli when it has one of the failures defined in the dictionary. The organization of this file is shown in Figure 4.10. First, the nominal circuit description is written. It is followed by the statements which modify the nominal circuit description to incorporate a failure mode. Then, circuit equivalent descriptions of the stimulus and measurement devices are written. The type of analysis (which is a function of the measurement and test strategy) is included. It is followed by an identification of the input which is used to identify the results in the circuit analysis output file. Finally the "run" instruction to simulate the above circuit description is given.

For each candidate test, all of the failure modes which are in the failure dictionary are put into this file one after another.

| | | |
|------------------|---|---|
| *CIRCUIT | | start a new circuit description |
| : | } | nominal circuit description |
| : | | |
| : | | |
| RE12V 5 0 0 E 12 | } | stimulus description |
| : | | |
| : | | |
| *SAVE | | save circuit file for later use |
| *DC *WORST VN1 | | circuit analysis type and quantity to be measured |
| :* 1 1 1 | | stimulus, measurement, failure mode(nominal) |
| *RUN | | start analysis |
| *LOAD | | copy back original circuit and stimulus |
| .R1 = 1.0E9 | } | failure definition changing the nominal circuit |
| : | | |
| : | | |
| *DC *WORST VN1 | | analysis type, measurement |
| :* 1 1 2 | | stimulus, measurement, failure no. 2 |
| *RUN | | |
| : | | |
| : | | |
| : | | |

Figure 4.10 NAP2 Candidate Tests Library Organization

4.1.7 NOPAL Candidate Tests Library

In addition to the candidate tests library which is written in the NAP2 language, the same test specifications are produced in NOPAL language and placed in another file (see Figure 4.11). This file contains partial test modules including only the stimulus, measurement, and message statements for a test. The remaining parts of a test specification (assertion and logic sections) are filled later when the final NOPAL output is being produced. Only the useful tests appear in the final output.

4.2 Failure Simulation and Symptom Generation

During this process, the behavior of the UUT is simulated with the circuit analysis program NAP2 (P3) for the test strategies, and for each one of the failure modes, and the nominal mode as enumerated in the NAP2 candidate tests library. The results of NAP2 simulation are saved in a permanent file. This file is read by the failure symptom generator program (P4). Then, the failure symptom table is produced. These two programs are described below.

4.2.1 Computer Aided Network Analysis

A computer aided network analysis (CANA) program is a computer aided design system which enables the user to simulate the behavior of electronic circuits. A CANA program uses the topological description and the component

```

/* TEST MODULE ( 2, 1, 1) - CC STRATEGY *****/
TEST 1 ;

STIMULI 1( 1) ;
/* THIS IS AN IMPEDENCE MEASUREMENT TEST - NO STIMULUS IS USED */

MEASUREMENT 2( 1) ;
CONJUNCTION ( 2) :
( < J1_B , J1_A > OHM = OHMMETER ( ZJ1_B_1 OHM ) )
TARGET : ZJ1_B_1 ;

/* TEST MODULE ( 22, 4, 24) - USER DEFINED TEST *****/
TEST 24 ;

STIMULI 47( 24) ;
CONJUNCTION ( 47) :
( < J1_G , J1_B > AMP = JSUPPLY ( 1.0E-6 AMP , 1.0E-6 MHO ) ;

MEASUREMENT 48( 24) ;
CONJUNCTION ( 48) :
( < J1_G , J1_B > VOLT = VOLTMETER ( VJ1_G3 VOLT ) ) ;
TARGET : VJ1_G3 ;

/* TEST MODULE ( 1, 1, 29) - DC STRATEGY *****/
TEST 29 ;

STIMULI 57( 29) ;
CONJUNCTION ( 57) :
( < J1_X , J1_B > VOLT = ESUPPLY ( 24.0 VOLT , 0.01 OHM ) ) &
( < J1_F , J1_B > VOLT = ESUPPLY ( 15.0 VOLT , 0.01 OHM ) ) &
( < J1_G , J1_B > VOLT = ESUPPLY ( 29.0 VOLT , 0.01 OHM ) ) &
( < J1_H , J1_B > VOLT = ESUPPLY ( 24.0 VOLT , 0.01 OHM ) ) ;

MEASUREMENT 58( 29) ;
CONJUNCTION ( 58) :
( < J1_A , J1_B > VOLT = VOLTMETER ( VJ1_A_29 VOLT ) )
TARGET : VJ1_A_29 ;

MESSAGE ( 29) : "THIS IS A SIMPLE TEST OF POWERING THE UUT WITH NOMINAL

```

Figure 4.11 NOPAL Candidate Tests Library

values of a circuit to formulate network equations which are then solved by numerical methods. CANA programs provide solutions in steady state, time domain, and frequency domain. Additional analysis capabilities may include network sensitivity, worst-case, Monte Carlo analysis and optimal design. Most of the CANA programs are implemented in the FORTRAN programming language. EXTENDED SCEPTRE [77], SPICE2 [78] and NET-II [79] are some of the circuit analysis programs which have a large number of analysis capabilities.

In this implementation of the FITS system, the NAP2 circuit analysis program [67] is used. The NAP2 program is incorporated into the FITS system without any internal changes. This choice makes it possible to use the most recent version of the program without any changes in the FITS system. It is possible to change this circuit analysis program with another by simply making minor syntactic changes in the candidate tests generator and failure symptom generator programs.

NAP2 program was selected because it can be executed interactively on the computers where the FITS system is developed. It has fast response time and provides the options needed by the FITS systems in a convenient fashion.

The input to the circuit analysis program is the NAP2 candidate tests library file. The output of the circuit analysis is stored in a permanent file for later reference.

4.2.2 Generation of the Failure Symptom Table

Because the NAP2 program is not internally modified, its output also contains irrelevant information for the purposes of fault isolation. The output of the simulation is scanned by the failure symptom generation program (P4) to extract only the information needed in the failure symptom table. The organization of this table (Table 4.5) is explained in the next section. If any numerical analysis problems are encountered, the user is warned about their existence. It is the user's responsibility to remove the conditions which cause numerical difficulties.

4.2.3 Failure Symptom Table

The failure symptom table is generated after a test is simulated for all failure modes of the UUT. It contains the following information (see Table 4.5): (1) stimulus identification, (2) measurement identification, (3) name of the quantity measured (target variable), and (4) the range of values measured at the indicated test terminal when the UUT has a failure. Six measurements are available after circuit simulation. They are tabulated in the following sequence: (1) nominal measurement, (2) minimum worst case, (3) maximum worst case, (4) nominal root sum square (RSS) sensitivity, (5) RSS sensitivity at minimum worst case, and (6) RSS sensitivity at maximum worst case.

The possible range of measurements at a specified test terminal of a UUT with a known failure mode is called a failure symptom.

Table 4.5 shows a partial listing from the failure symptom table of the CPS example. The complete table is about 10 pages long. Since this is an internal data structure transparent to the user, it is not described further.

TABLE 4.5

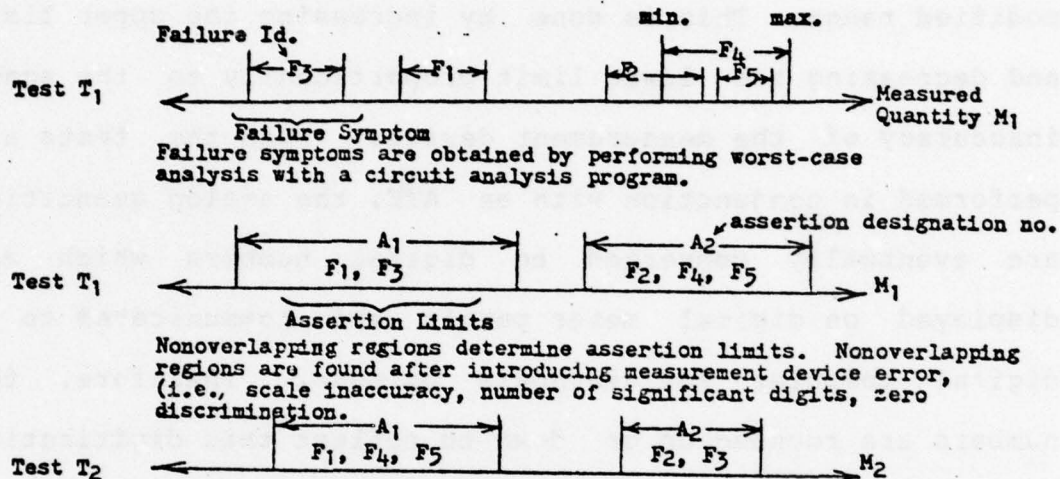
| STIM | REAS | FAIL | TARGET | NOMINAL | MINIMUM | MAXIMUM | NOM. SENS. | MIN. SENS. | MAX. SENS. |
|------|------|------|--------|----------|---------|----------|------------|------------|------------|
| 1 | 2 | 2 | W | 2.778950 | 01 | 2.794534 | 01 | 1.462680 | 00 |
| 1 | 2 | 3 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 4 | W | 2.778950 | 01 | 2.794534 | 01 | 1.462680 | 00 |
| 1 | 2 | 5 | W | 2.437710 | 01 | 2.436458 | 01 | 1.564717 | -02 |
| 1 | 2 | 6 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 7 | W | 2.778950 | 01 | 2.794534 | 01 | 1.462680 | 00 |
| 1 | 2 | 8 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 9 | W | 2.778950 | 01 | 2.794534 | 01 | 1.462680 | 00 |
| 1 | 2 | 10 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 11 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 12 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 13 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 14 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 15 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 16 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 17 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 18 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 19 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 20 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 21 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 22 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 23 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 24 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 25 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 26 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 27 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 28 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 29 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 30 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 31 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 32 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 33 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 34 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 35 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 36 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 37 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 38 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 39 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 40 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 41 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 42 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 43 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 44 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 45 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 46 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 47 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 48 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 49 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 50 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 51 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 52 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 53 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 54 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 55 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 56 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |
| 1 | 2 | 57 | W | 2.635710 | 01 | 2.634550 | 01 | 1.564717 | -02 |
| 1 | 2 | 58 | W | 2.437710 | 01 | 2.436458 | 01 | 1.655377 | -02 |

4.3 Decision Limit Determination and Ambiguity Analysis

The two processes in this section determine easily recognizable decision limits and calculate the level of fault isolation attainable using these limits. These two processes are described below.

4.3.1 Decision Limit Finder

The process P5 involves finding regions of normal and malfunctioning operation. The inputs to the process are the failure symptom table and the accuracy specifications. The two outputs produced are the assertion limit and binary valued decoder table, and the test limit and multiple valued decoder table. Figure 4.12 illustrates how this process combines similar failure symptoms together to find easily recognizable ranges of measurement to identify groups of failures. The minimum and maximum limits of values of measurements at the test terminals (symptoms) of each failure due to a stimulus, as found by circuit simulation, are not exactly what a measurement device would show because the ranges are initially calculated to a relatively high number of significant digits (typically six). It is unreasonable to expect such high accuracy from most test equipment. In order to allow measurement tolerance (note this is different from loading effects), the analog and digital natures of measurement devices are considered.



Tests 1 and 2 as shown above determine the following decoder matrix.

| | | | | | | |
|------------------|----------------|----------------|----------------|----------------|----------------|--|
| | F ₁ | F ₂ | F ₃ | F ₄ | F ₅ | |
| T ₁ : | 1 | 2 | 1 | 2 | 2 | ← Diagnosis selection by disjunctions Diagnosis selection by conjunctions |
| T ₂ : | 1 | 2 | 2 | 1 | 1 | |

Observe that F₁, F₂, F₃ can be isolated by these two tests; however it is not possible to distinguish between F₄ and F₅.

Figure 4.12 Pictorial Representation of Regions

First very small measurement limits below a threshold level are eliminated (zero discrimination). The reason for this was explained in Section 4.1.1.4. Then, each measurement limit is adjusted so that whatever the measurement device shows because of the "scale inaccuracy" is covered in the modified range. This is done by increasing the upper limit and decreasing the lower limit proportionally to the scale inaccuracy of the measurement device. When the tests are performed in conjunction with an ATE, the analog quantities are eventually converted to digital numbers which are displayed on digital meter panels and communicated to a digital computer for diagnosis purpose. Therefore, the numbers are rounded up or down to reflect this digitization effect. Most digital meters have better than 3-digit accuracy.

After these initial adjustments are made, the algorithm proceeds to find the extent of regions so that they do not overlap. The upper and lower limits of each region are used to generate an assertion. Thus a measurement, depending on the stimulus, failures and UUT, may yield from 1 to N assertions, where N is the number of failure modes contained in the failure dictionary. If there is only one assertion, it means that all symptoms overlap and the measurement is totally useless for diagnostic purposes. If there are N assertions (which is unlikely to happen), searching for more tests may stop right there, since there will be one assertion to isolate each failure mode. The assertions are numbered

sequentially as they are created. The sequence numbers are called assertion designation numbers. They are used to identify the assertion limits.

After the assertion limits are found, the failure symptoms which fall into these ranges are identified in the failure decoder section of the Assertion Limit and Binary Valued Decoder Table (Table 4.6). The rows of this table are identified by the stimulus, measurement and an assertion designation number. The columns are labelled by the failure mode sequence numbers. If a failure has its symptom in the specified range of an assertion, then the entry in the corresponding column is "1" indicating "true". If it is not in this range, then the entry is "0" indicating "false".

The primary purpose of the test limit and multiple valued decoder table is to provide a summary of the longer binary valued diagnosis table. It also helps to save computer time during the optimization processes.

In this process, the user has several options to control and experiment with the assertions created. One of these options effectively deletes all of the symptoms of a failure from consideration. Another option makes it possible to equivalence failures by overriding the simulation results. The implications of these modifications are problem dependent. They are provided to aid the user to understand how important are some failure symptoms in determining the fault isolation level.

Another option makes it possible to list the assertions and

AD-A056 660

MOORE SCHOOL OF ELECTRICAL ENGINEERING PHILADELPHIA P--ETC F/G 9/3
AUTOMATED TEST DESIGN.(U)

JUN 78 C TINAZTEPE

DAAA25-75-C-0650

UNCLASSIFIED

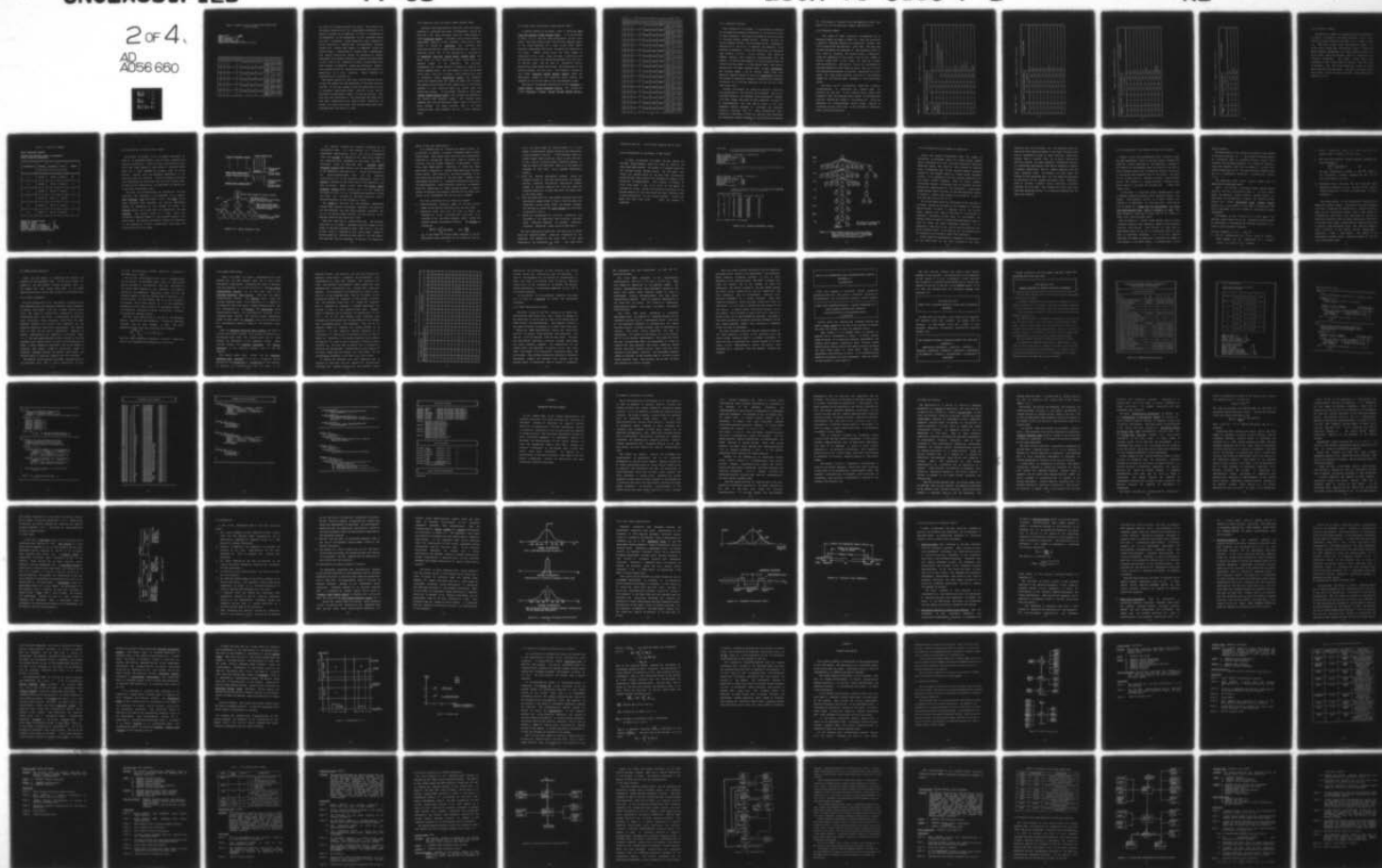
77-03

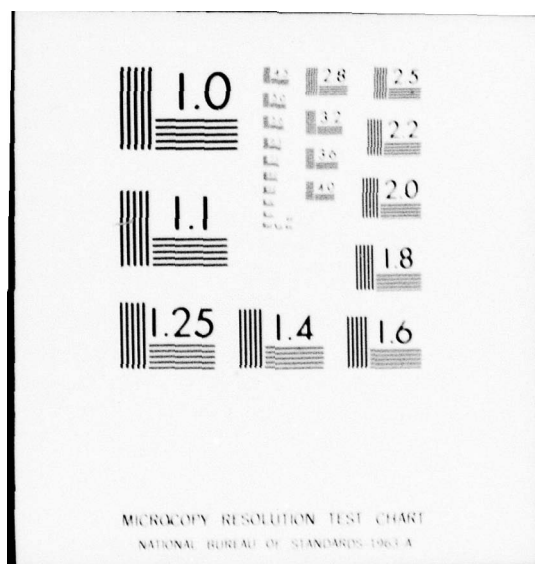
ECOM-75-0650-F-2

NL

2 of 4

AD
A056 660





the tests in increasing sensitivity order. The average root sum square sensitivity of the measurements belonging to the group of failures in an assertion (or test) is indicative of the dependence of the measurement on the component values and their tolerances. Larger sensitivity when compared to smaller sensitivity implies that the measurement obtained changes more rapidly with respect to component values and failure modes. Intuitively, it appears that a measurement with smaller sensitivity should be preferred to another measurement with higher sensitivity because the measurement in this case is not expected to change significantly with small changes in component values and tolerances. However, this argument cannot be generalized because sensitivity, by definition, is a local behavior. These concepts are explained more in Section 5.5.

The default options to test limit finding program are set so that the tables list the assertions in the order they are created. If the user wishes to sort the assertions or tests in increasing sensitivity order, then one of the sorting strategies as described in Table 4.3 may be selected. FITS then will give preference to the assertions and tests which have lower sensitivity to be used in fault isolation. Of course, its likely that there will be some additional tests performed when this option is used.

4.3.2 Assertion Limit and Binary Valued Decoder Table

Decision limit determination algorithm uses the failure symptoms to partition the range of measurement values for each test. The lower and upper limits of these ranges are called assertion limits. A statement which checks whether a measurement taken at a test terminal falls within these limits is called an assertion. The algorithm also identifies the failure modes whose symptoms lie within the assertion limits. All of this information is displayed in the assertion limit and binary valued decoder table (see Table 4.6). In this table, the first column shows the sequence number of the assertion. The stimulus, measurement, and target variable columns are same as the failure symptom table. For each assertion which can be made from a test, a new row is created. Each assertion of a test is assigned a unique designation number. The assertion limits are entered under the lower and upper limit columns. The average root-sum-square (RSS) sensitivity of the failure symptoms in this assertion range are placed under the sensitivity column. In the decoder section of this table (binary valued decoder matrix), the columns are labelled by the failure mode sequence number. The failures whose symptoms fall into the assertion ranges have a "1" entry in their columns. All other failures have a "0" entry indicating that their symptoms are not in this assertion range.

4.3.3 Test Limit and Multiple Valued Decoder Table

A smaller version of the above table is called the test limit and multiple valued decoder table. As it can be seen in Table 4.7, it has the same organization as the first table. The first column gives the starting sequence number of the first assertion of a test in the first table. Stimulus, measurement and target variables are identical to the first. "DESIG" column gives the total number of assertions of this test. Lower and upper limits given are the absolute minimum and maximum measurements for this test. If desired, they may be used as protection limits. Sensitivity column shows the average RSS sensitivity of all failure symptoms for this test. The diagnosis section of this table (multiple valued decoder matrix) shows the designation number of the assertion which contain the symptoms of the failure mode of the corresponding column.

The rows of the decoder matrices are called (multiple or binary valued) failure diagnosis vectors. The columns are called (multiple or binary valued) failure address vectors.

TABLE 4.7 TEST LIMIT AND MULTIPLE VALUED DECODER TABLE

| SSO | STIR | PLAS | TARGET VARIABLE | DESIG | LOWER LIMIT | UPPER LIMIT | SENSITIVITY | MULTIPLE VALUED DECODER MATRIX |
|-----|------|------|-----------------|-------|--------------|--------------|-------------|--|
| 1 | 2 | 1 | 2u 1 | 6 | 9.97999E-02 | 0.44000E 07 | 4.00000E-01 | 3 |
| 3 | 3 | 2 | 2u 2 | 10 | 4.00000E 03 | 9.79000E 07 | 3.03700E 01 | 0 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 3 1 1 0 1 0 1 0 4 2 0 0 0 0 0 0 |
| 15 | 4 | 3 | 2u 3 | 7 | 5.29700E-02 | 9.99999E 00 | 1.97507E 02 | 1 |
| 22 | 5 | 4 | 2u 4 | 8 | 5.50000E 02 | 2.30000E 05 | 4.30470E 02 | 0 |
| 32 | 15 | 4 | vu 4 | 4 | 0.00000E 00 | 5.92000E-03 | 1.79002E-05 | 1 |
| 34 | 15 | 2 | vu 2 | 11 | -9.91000E-01 | 0.00000E 00 | 2.10002E-04 | 1 |
| 45 | 22 | 4 | vu 4 | 7 | -1.07400E-01 | 0.00000E 00 | 4.30452E-04 | 1 |
| 52 | 20 | 1 | vu 1 | 5 | 0.00000E 00 | 2.57200E-03 | 4.57949E-07 | 1 |
| 55 | 27 | 4 | vu 4 | 7 | 0.00000E 00 | 1.00000E-01 | 4.13572E-04 | 1 |
| 62 | 28 | 5 | vu 5 | 5 | -2.45000E-01 | 0.00000E 00 | 4.50000E-04 | 2 |
| 67 | 31 | 5 | vu 5 | 10 | -2.79200E-01 | 0.00000E 00 | 4.20002E-04 | 4 |
| 77 | 1 | 1 | vu 1 | 4 | 2.37000E 01 | 3.00000E 01 | 0.00000E 00 | 1 |
| 81 | 1 | 2 | vu 2 | 4 | 1.00000E 01 | 3.00000E 01 | 4.35704E-01 | 3 |
| 85 | 1 | 3 | 1 0015 | 4 | -4.12500E-02 | 4.12500E-02 | 5.95527E-04 | 3 |
| 89 | 1 | 4 | 1 0020 | 4 | -1.51000E 02 | -2.49100E-02 | 1.00520E-01 | 4 |
| 93 | 1 | 5 | 1 0024 | 6 | -2.90000E 02 | 0.00000E 00 | 5.90510E-05 | 5 |
| 99 | 1 | 6 | 1 0007 | 4 | 0.00000E 00 | 5.00000E 01 | 1.12274E-02 | 2 |

4.3.4 Ambiguity Analysis

After Process P5 is finished, it is possible to evaluate all accumulated diagnosis information to determine the level of fault isolation. The input to Process P6 can be a binary or multiple valued decoder matrix. Process P6 uses the failure address vectors of the decoder matrix described in Sections 4.3.2 and 4.3.3 to evaluate the ambiguity level achieved to determine if the test objectives are satisfied (see Section 4.1.1.4). If two failures have identical address vectors, they cannot be distinguished from one another. If one of them happens to be the nominal mode (by notation failure number 1 is the nominal mode), then another failure with the same address vector is not diagnosable. The relation among a set of failure modes which have identical address vectors is called an equivalence relation. The concept of equivalent classes of failures was introduced in Section 4.1.1.4 and was explained with the aid of a graph (see Figure 4.5).

Process P6 performs an ambiguity analysis indicating what percentage of the failures are diagnosed, and how the diagnosed failures are isolated into equivalence classes. If at this stage, the level of fault isolation is found to be unsatisfactory, the next set of tests of the next candidate testing strategy is initiated and evaluated similarly, starting from P3. After satisfactory fault isolation is achieved, or after all possible test strategies are exhausted, control transfers to the optimization process

P7. The outputs of Process P6 are the ambiguity report (see Table 4.8), and the ambiguity summary (see Table 4.9).

4.3.5 Ambiguity Report

The result of fault isolation is presented as an ambiguity report as shown in Table 4.8. Each row identifies a group of failures isolated into equivalent classes. The first column gives the ambiguity class name. The same name is used throughout the remainder of the outputs to refer to this class of failures. The second column, k-ambiguity, gives the ambiguity of this class; that is, the number of failure modes found in the class. The next two columns provide statistical information. First, the percentage of failures isolated in this class is given. Next to it is the cumulative percentage of failures identified up to this class. The last column contains the list of the sequence numbers of the failure modes isolated in this equivalence class.

The first row of the ambiguity report has a special interpretation. It identifies the nominal mode (no failures) of the UUT. If the testing cannot assure 100% diagnosis, then the failures which cannot be distinguished from the nominal are listed in the second row. Since the existence of nondiagnosable failure modes implies no isolation as well, this class is not included in cumulative fault isolation percentage.

TABLE 4.8 AMBIGUITY REPORT

| EQUIVALENCE CLASS | K-AMBIGUITY | FAULT ISOLATION PERCENTAGE | | FAILURE MODES INCLUDED |
|----------------------|-------------|----------------------------|------------|---|
| | | CLASS | CUMULATIVE | |
| NOMINAL | 1 | 1.5% | 1 | |
| NO-DIAGNOSIS | 13 | 19.4% | | 1, 3, 5, 6, 13, 15, 19, 26, 57, 59, 63, 65, 67, |
| 2 | 10 | 14.9% | 14.9% | 2, 4, 7, 9, 34, 35, 36, 38, 47, 60, |
| 3 | 1 | 1.5% | 16.4% | 8, |
| 4 | 1 | 1.5% | 17.9% | 10, |
| 5 | 1 | 1.5% | 19.4% | 11, |
| 6 | 2 | 3.0% | 22.4% | 12, 62, |
| 7 | 1 | 1.5% | 23.9% | 14, |
| 8 | 3 | 4.5% | 28.4% | 16, 40, 41, |
| 9 | 4 | 6.0% | 34.3% | 17, 20, 61, 64, |
| 10 | 1 | 1.5% | 35.8% | 18, |
| 11 | 2 | 3.0% | 38.8% | 21, 66, |

TABLE 4.8 AMBIGUITY REPORT (Continued)

| EQUIVALENCE CLASS | K-AMBIGUITY | FAULT ISOLATION PERCENTAGE | | FAILURE MODES INCLUDED |
|----------------------|-------------|----------------------------|------------|------------------------|
| | | CLASS | CUMULATIVE | |
| 12 | 1 | 1.5% | 40.3% | 22, |
| 13 | 1 | 1.5% | 41.8% | 23, |
| 14 | 1 | 1.5% | 43.3% | 24, |
| 15 | 1 | 1.5% | 44.8% | 25, |
| 16 | 1 | 1.5% | 46.3% | 27, |
| 17 | 2 | 3.0% | 49.3% | 28, 37, |
| 18 | 1 | 1.5% | 50.7% | 29, |
| 19 | 1 | 1.5% | 52.2% | 30, |
| 20 | 1 | 1.5% | 53.7% | 31, |
| 21 | 1 | 1.5% | 55.2% | 32, |

TABLE 4.8 AMBIGUITY REPORT (Continued)

| EQUIVALENCE CLASS | K-AMBIGUITY | FAULT ISOLATION PERCENTAGE | | FAILURE MODES INCLUDED |
|----------------------|-------------|----------------------------|------------|------------------------|
| | | CLASS | CUMULATIVE | |
| 22 | 1 | 1.5% | 56.7% | 33, |
| 23 | 1 | 1.5% | 58.2% | 39, |
| 24 | 1 | 1.5% | 59.7% | 42, |
| 25 | 1 | 1.5% | 61.2% | 43, |
| 26 | 1 | 1.5% | 62.7% | 44, |
| 27 | 2 | 3.0% | 65.7% | 45, 46, |
| 28 | 1 | 1.5% | 67.2% | 48, |
| 29 | 1 | 1.5% | 68.7% | 49, |
| 30 | 3 | 4.5% | 73.1% | 50, 51, 52, |
| 31 | 1 | 1.5% | 74.6% | 53, |

TABLE 4.8 AMBIGUITY REPORT (Continued)

| EQUIVALENCE CLASS | K-AMBIGUITY | FAULT ISOLATION PERCENTAGE | | FAILURE MODES INCLUDED |
|----------------------|-------------|----------------------------|------------|------------------------|
| | | CLASS | CUMULATIVE | |
| 32 | 1 | 1.52 | 76.12 | 54, |
| 33 | 2 | 3.02 | 79.12 | 55, 56, |
| 34 | 1 | 1.52 | 80.62 | 58, |

4.3.6 Ambiguity Summary

The ambiguity report is organized slightly differently to present a summary of the fault isolation level. Fault isolation summary obtained from Table 4.8 is shown in Table 4.9. The first column gives the ambiguity level. The second column shows what the desired level of cumulative fault isolation percentage is for this ambiguity level. The third column shows the achieved level of cumulative fault isolation percentage. The fourth column shows what percentage of the total number of failures have this level of ambiguity. The last column gives the number of classes which have the same ambiguity. The fault isolation summary may be used to plot the fault isolation curve described in Section 4.1.1.4.

TABLE 4.9 AMBIGUITY SUMMARY

FAULT ISOLATION SUMMARY

DESIRED AND ACHIEVED LEVEL OF CUMULATIVE
FAULT ISOLATION PERCENTAGE

| K-AMBIGUITY | DESIRED C.F.I.P. | ACHIEVED C.F.I.P. | CLASS F.I.P. | NUMBER |
|-------------|---------------------|----------------------|-----------------|--------|
| 1 | 25.0% | 35.8% | 35.8% | 24 |
| 2 | 50.0% | 50.7% | 14.9% | 5 |
| 3 | 50.0% | 59.7% | 9.0% | 2 |
| 4 | 50.0% | 65.7% | 6.0% | 1 |
| 5 | 60.0% | 65.7% | 0.0% | 0 |
| 10 | 80.0% | 80.6% | 14.9% | 1 |

NUMBER OF FAILURE MODES : 67
 NUMBER OF TESTS : 17
 NUMBER OF EQUIV. CLASSES : 34
 DESIRED LEVEL OF DIAGNOSIS: 80.0%
 ACHIEVED LEVEL OF DIAGNOSIS: 80.6%
 FAULT ISOLATION IS SATISFACTORY.

4.4 Optimization to Minimize Test Length

The purpose of Process P7 is to reduce the number of tests to be performed without loss of fault diagnosis and isolation capability. There are three optimization steps. First, the total number of test setups is minimized since setup for a test consumes the longest time in actual testing. Second, only the necessary assertions of the remaining tests are retained. Finally, diagnosis selection logic statements are found such that only the minimum number of tests in the remaining set are performed to diagnose and isolate each equivalence class.

The first two considerations are attained by the same algorithm. The approach in the algorithm is to create a fault isolation tree (see Figure 4.13). The nodes of the fault isolation tree represent the failure equivalence classes as determined by the outcomes of tests. The branches coming out of each node are labelled by the test outcomes. Each branch links to a lower level node containing a disjoint subset of the failures present in the parent node. The smaller equivalence classes are determined by the restriction of test outcome which also names the branch pointing to this node.

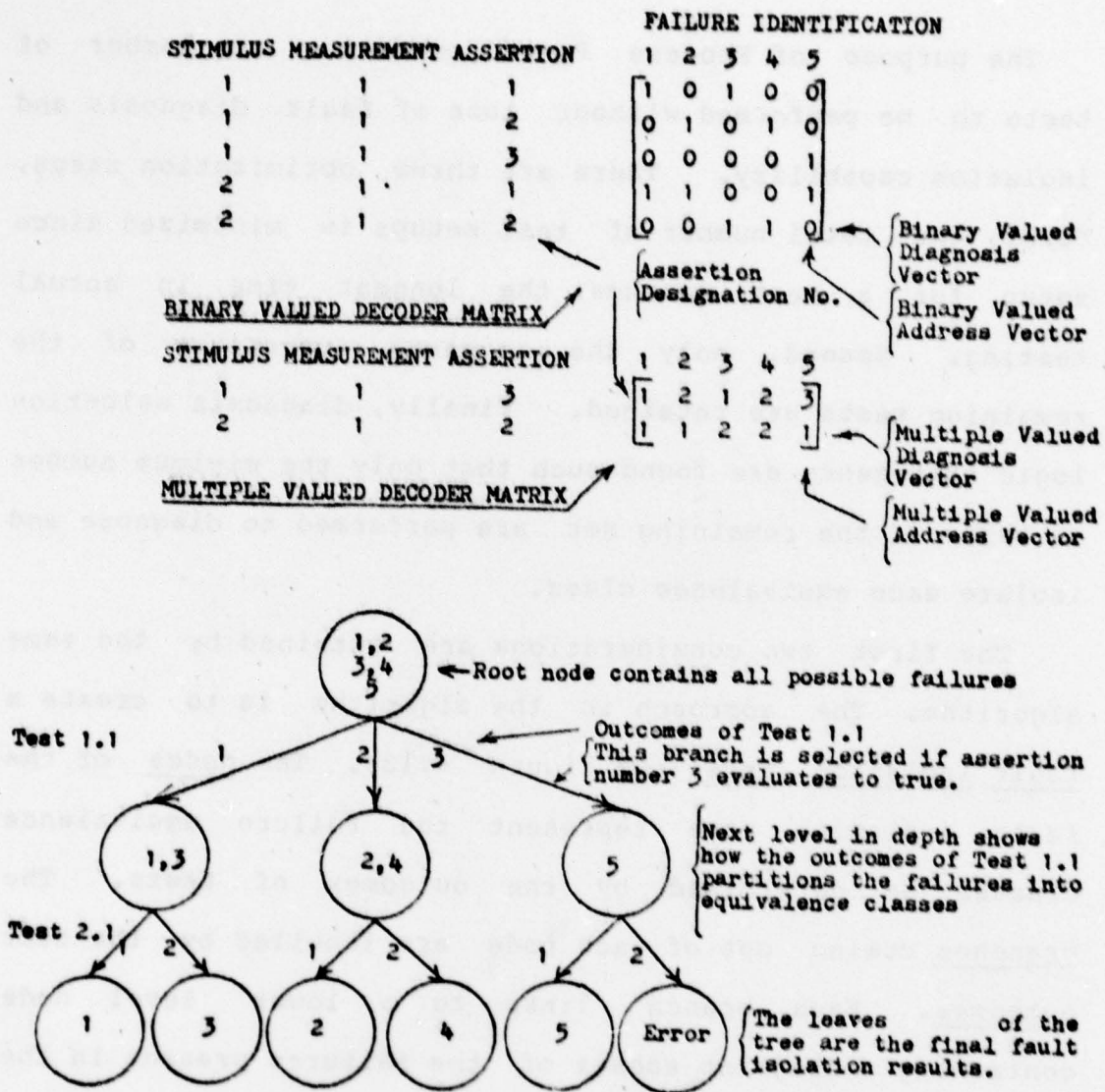


Figure 4.13 Fault Isolation Tree

Two types of outcomes are possible depending on the optimization step. If a test consists of a stimulus and measurement specification (as in the setup minimization step), the outcome is defined as the designation number of the assertion which evaluates to true after the test is performed. Tests of this type have multiple valued diagnosis vectors whose entries are the designation numbers of the assertions (see Table 4.7). If the test consists of one stimulus-measurement pair and one assertion specification (as in the assertion minimization step), the outcome is defined as the truth value of the assertion (i.e., true or false). Tests of this type have binary valued diagnosis vectors whose entries are 1 if the assertion evaluates to true, or 0 if the assertion evaluates to false (see Table 4.6). A branch of the fault isolation tree is selected depending on the outcome of a test.

The leaves of the tree are the resultant equivalence classes after testing is finished. Therefore, the leaves of the tree indicate the final fault isolation achieved. Any one of the failure modes included in each leaf may be the possible cause of the failure of a UUT.

There are many ways to create this tree. The approach taken here is to create a balanced tree with respect to the number of failures included in each node; that is, the test at each level is selected so that nearly equal number of failure modes are isolated at each node of the next depth. This approach has the advantage of avoiding an exhaustive

search in the tree construction.

It is assumed that all failures are equally likely. It is possible to assign a relative frequency index to each failure mode. This option would force the tree construction algorithm to include the tests which tend to isolate the failures with the higher frequency of occurrence earlier (i.e., with fewer tests). Therefore, if there are tests which make it possible to isolate these particular failures before others, such tests will be given precedence in the tree (they will be closer to the root node). However, this requirement may result in introducing additional tests into the specifications, which otherwise would not be present. Since the bottom part of NOPAL already provides a similar option to optimize the execution sequence of the tests, this option is not implemented into the top part of NOPAL.

The fault isolation tree is created as follows:

1. [Initialization] Place the names of all the failure modes of the failure dictionary into the root node.
2. [Determine which test will be used to define the branches out of the root node] The first test to be accepted is the one whose diagnosis vector yields the highest entropy (information content). The entropy is defined as:

$$H(T_i) = \sum_{j=1}^n p_{ij} \log p_{ij} \quad ; \quad p_{ij} = \frac{k_{ij}}{N_F}$$

k_{ij} is the number of failure modes diagnosed in the j^{th} equivalence class determined by the outcome of test T_i .

N_F is the total number of failure modes; "n" is the number of equivalence classes formed by the restriction of the outcomes of test T_i . If the diagnosis vector is binary valued, then a node can split at most into two. In the case of multiple valued diagnosis vectors, a node can split to m new nodes where m is the number of outcomes of that test (i.e., maximum designation number).

3. [Find the smaller equivalence classes] Using the diagnosis vector of the test which yields the highest entropy, construct the distinct equivalence classes (nodes) at the next depth of the tree and label the branches coming out of each node from the upper level to the lower level.
4. [Find the next test with the highest entropy] Given the equivalence classes found in Step 3, find the next test (diagnosis vector) which gives the highest entropy. This entropy is called the joint entropy and calculated as defined in Step 2.
5. [Check for termination] If the joint entropy has not increased or all tests have been utilized, then best fault isolation possible is reached, therefore, terminate. Otherwise, repeat starting from Step 3.

The tree construction stops when the tests can no longer split the remaining nodes. After the termination of the algorithm, the subsets of the tests used in the tree construction are sufficient to reach the same fault

isolation level as the original complete set of tests.

4.4.1 Minimization of the Number of Test Setups

In order to minimize the number of test setups, the multiple valued decoder matrix is given as input to the above described program. The output is a reduced version of the multiple valued decoder matrix. As the reader will recall, the decoder matrix of the test limit and multiple valued decoder table contains the designation numbers of the assertions of measurements. This results in forming an n-ary tree, where n may be as large as the number of outcomes of a test. Figure 4.14 shows the evaluation of the entropy measure for the tests while the fault isolation tree is being constructed for the UUT_CPS example. Figure 4.15 shows how this tree looks after the process is completed.

FAILURE MODES 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67

TEST (5) STIMULUS (3) MEASUREMENT (2) ASSERTION (10)
 NUMBER OF TESTS USED : 1
 NUMBER OF EQUIVALENCE CLASSES : 10
 ENTROPY REQUIRED FOR ISOLATION : 0.066
 CURRENT ENTROPY : 1.286
 CAPACITY (THEORETICAL MAX.) : 3.322
 EQUIVOCATION : 4.780
 EFFECTIVENESS : 0.387
 MEMBERSHIP IN EQUIV. CLASSES

```

      1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 3 4 1
      1 1 1 4 1 1 5 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 6 7 8
      6 8 9 10 1 1 1 1 1 1 1 1 1 1 1 1 1
      .
      .
      .
      .
  
```

TEST (87) STIMULUS (1) MEASUREMENT (4) ASSERTION (4)
 NUMBER OF TESTS USED : 8
 NUMBER OF EQUIVALLNCE CLASSES : 34
 ENTROPY REQUIRED FOR ISOLATION : 0.066
 CURRENT ENTROPY : 4.442
 CAPACITY (THEORETICAL MAX.) : 5.087
 EQUIVOCATION : 1.624
 EFFECTIVENESS : 0.873
 MEMBERSHIP IN EQUIV. CLASSES

```

      1 2 1 2 1 1 2 3 2 4 5 6 1 7 1 8 9 10 1 9 11 12 13 14 15
      1 16 17 18 19 20 21 22 2 2 2 17 2 23 8 8 24 25 26 27 27 2 28 29 30
      30 30 31 32 33 33 1 34 1 2 9 6 1 9 1 11 1
  
```

INDIVIDUAL AND JOINT ENTROPY VALUES

| SEQ | STIM | MEAS | ASSE | ENTROPY | JOINT ENTROPY | EQUIV. CLASS |
|-----|------|------|------|---------|---------------|--------------|
| 2 | 3 | 2 | 5 | 1.286 | 1.286 | 10 |
| 4 | 5 | 4 | 22 | 1.093 | 2.290 | 18 |
| 3 | 4 | 3 | 15 | 0.815 | 3.027 | 24 |
| 13 | 1 | 2 | 79 | 1.270 | 3.637 | 26 |
| 17 | 1 | 6 | 96 | 0.654 | 3.985 | 29 |
| 6 | 15 | 2 | 34 | 1.231 | 4.204 | 31 |
| 10 | 28 | 5 | 61 | 0.446 | 4.364 | 33 |
| 15 | 1 | 4 | 87 | 0.416 | 4.442 | 34 |
| 1 | 2 | 1 | 1 | 0.416 | 4.442 | 34 |
| 5 | 15 | 4 | 30 | 0.497 | 4.442 | 34 |
| 7 | 22 | 4 | 44 | 1.063 | 4.442 | 34 |
| 8 | 28 | 1 | 51 | 0.223 | 4.442 | 34 |
| 9 | 31 | 4 | 54 | 1.165 | 4.442 | 34 |
| 11 | 31 | 5 | 66 | 1.044 | 4.442 | 34 |
| 12 | 1 | 1 | 75 | 0.497 | 4.442 | 34 |
| 14 | 1 | 3 | 83 | 0.335 | 4.442 | 34 |
| 16 | 1 | 5 | 91 | 0.526 | 4.442 | 34 |

OUT OF 17 CANDIDATE TESTS 8 ARE RETAINED

Figure 4.14 Entropy Evaluation Output

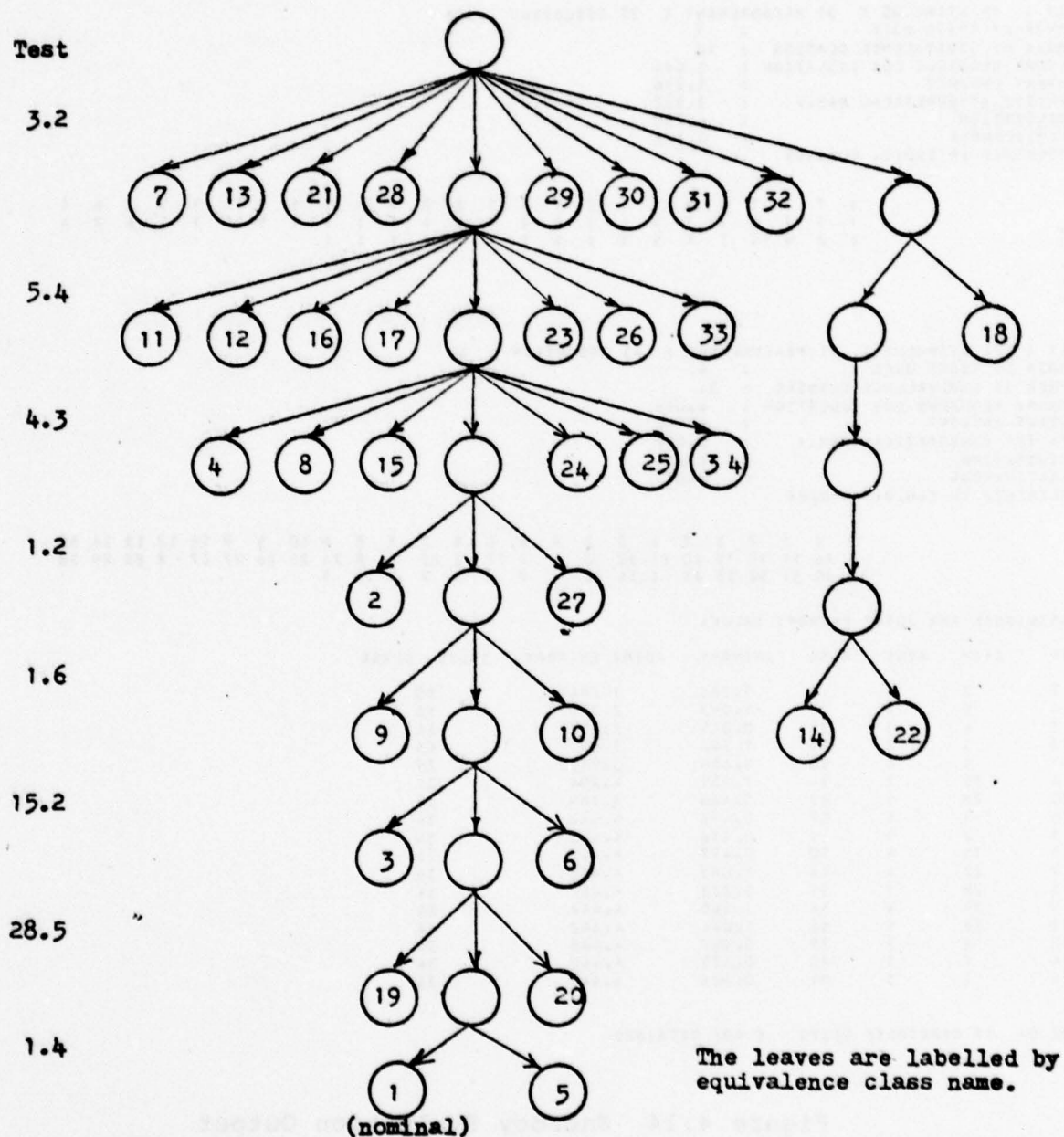


Figure 4.15 Fault Isolation Tree for the UUT-CPS Example (after first optimization with multiple valued decoder matrix) See Table 4.8 to find the members of the equivalence classes.

4.4.2 Minimization of the Number of Assertions

During the second minimization step, the number of assertions is minimized using the binary valued decoder matrix (see Table 4.6) with the tree construction algorithm described above. The input to this process is obtained after transforming each of the multiple valued diagnosis vectors in the reduced matrix to binary valued diagnosis vectors. It is possible to use the original assertion limit and binary valued diagnosis table as input; however, significant savings in computer time is achieved by going through two applications of this algorithm. Using binary valued decoder matrix as input to the same algorithm results in forming a binary tree, because the outcome of a test in this case may only be "0" or "1".

All of the tests which are referenced by the branches of the fault isolation tree need to be included in the NOPAL specifications in order to diagnose all of the failure modes. However, only some of them need to be performed to diagnose a particular failure mode. On the average, the minimum number of tests to isolate each equivalence class is found by using the above described algorithm. It is important to note that at the end of each minimization, the fault isolation level is the same as can be obtained using the results of all available tests from simulation.

During these two minimization processes, the assertions of the tests which do not form branches of the fault

isolation tree are eliminated. All the remaining tests and assertions which define branches in the fault isolation tree must be performed in order to make a decision about which failure mode to isolate. This is an overly restrictive condition because the same diagnostic information relating to a particular equivalence class may be repeated or implied in several assertions. This is readily observed with the fault isolation tree. For example, at some nodes the outcome of the referenced test will not cause any splitting to take place (for an example see Figure 4.15). Therefore, there is no need to conduct such a test when a specific diagnosis is to be made. The properties of this tree could have been exploited further if an execution sequence for the tests were desired.

4.4.3 Minimization of the Number of Tests per Diagnosis

Finally, as the third minimization step, a search is made for each equivalence class of failures (which contains the names and failure functions of the components reported to the operator as the probable cause of failure of the UUT) to find a minimum length join of assertions which is sufficient to distinguish one diagnosis from all the others. The input to the process can be a multiple or binary valued decoder matrix. The output is a sparse diagnosis selection logic matrix which is the basis for creating the NOPAL table (Table 4.10).

If the input is a multiple valued decoder matrix, then no negated diagnosis selection logic operators appear in the output. This approach results in a larger number test modules to be specified. In this case test limits become more stringent and demand that all measurements lie within some predetermined range before a diagnosis is made. This means diagnosis selection is based only on the passing of tests (i.e., using GO-paths only).

When the input is a binary valued decoder matrix, negated diagnosis selection logic operators appear in the diagnosis selection logic matrix. Thus, failing of tests (due to measurements which do not lie in a specified range) are also used in selecting diagnoses. Even though this approach minimizes the number of assertions (therefore the number of test modules in the NOPAL output), it becomes easier to make

false diagnosis.

The method used in this process is related to the concept of Boolean difference [80]. The Boolean Difference Method is a technique used in automatic digital circuit fault diagnosis and isolation programs. It is based on finding an input test sequence such that the output of a circuit will change states when one of the input variables changes state due to a catastrophic (stuck-at-1 or stuck-at-0) failure [81,82].

The algorithm which finds minimum length join of assertions is described below:

Let D be a binary or multiple valued decoder matrix obtained from the optimization Phase 2. In this matrix merge the identical columns (failure address vectors) so that there are no duplicate columns. The distinct column vectors are called equivalence class address vectors $a_j = f_j(t_1, t_2, \dots, t_n)$ where n is the number of assertions, and $j=1, 2, \dots, N_d$ where N_d is the number of equivalence classes (diagnoses).

The purpose of this algorithm is to find one of the shortest length substrings in the conjunction term a_j which is sufficient to distinguish the diagnosis selected by a_j from all other possible diagnoses.

For each diagnosis j , ($j=1, \dots, N_d$) do:

1. [For all combinations] Let S be a vector of length l whose members are the combination of l integers ($l=1, 2, \dots, n$) taken out of n integers.

2. [Select a substring] Using S to index a_j , find the substring d_j^S of length l of t_i 's in a_j .

3. [Find the first shortest unique substring] Evaluate the logic expression:

$$\prod_{i=1}^{N_d} (d_i^S \neq d_j^S) = 1 \quad ; \quad i \neq j$$

for each combination of length l . The \prod symbol is the conjunction (product) operator and \neq is the not-equal operator as usual.

4. [Expression found] The first d_i^S for which the above expression evaluates to true is used as the minimum length conjunction of tests (or assertions) to select diagnosis j .

The effectiveness of the minimization algorithms was checked for several cases. One example initially consisted of 17 test setups and 145 assertions to isolate 60 failure modes into 40 equivalence classes. Only 8 test setups and 20 assertions were found to be sufficient to achieve the same fault isolation level by these minimization algorithms. In the last step, about 3 assertions per diagnosis were used to isolate each equivalence class. The shortest diagnosis used 1 assertion (1 test setup) and the longest used 7 assertions (also 7 test setups). The output of the optimization process yields a modified version of the test limit and decoder table.

4.5 NOPAL Output Generation

After the test design is completed, the results are outputted in two different forms. (1) The first output is a summary of the test modules written in tabular form. (2) The second and last output from FITS is the NOPAL specification which can be used as input to the bottom part.

4.5.1 Output Processors

The final Process P8 in the test design is outputting the test specifications and diagnosis selection logic statements in tabular form (Table 4.10) and in NOPAL syntax (Figure 4.16). The input to Process P8 is prepared in the third minimization step. This input is essentially the assertion limit and binary valued diagnosis table which has been modified slightly to accommodate the "sparsity" and the diagnosis selection logic with conjunctions. The tabular form is used for the user's convenience. Examples to tabular representation has appeared in an earlier report [73]. The test specifications are also written in a string language to be directly given as input to the bottom part of NOPAL. A short summary of statistics such as the number of different test setups, number of assertions, number of diagnosis messages, minimum and maximum test length, and average number of tests per diagnosis are also provided.

The program which generates the NOPAL specifications is not described here. What it does is essentially to write

the test specifications in NOPAL language as described in the NOPAL user's guide.

The last two test modules in the string language output are not present in the NOPAL table. They are used to send special messages to the operator. The first one checks if the nominal mode diagnosis D_1 was selected. It is done by evaluating the expression $\prod_i (t_i = a_{i1})$ where i 's are the sequence numbers of the test modules used to select D_1 , t_i is the outcome of a test module and a_{i1} is the outcome in the decoder matrix which selects D_1 . If this expression evaluates to true, then the UUT is in the nominal state. If the expression evaluates to false, the UUT has a failure. An appropriate message follows.

The last test module finds if any one of the diagnoses made by conjunctions were selected. If none of them were selected, then an error message is sent. The logic expression which finds this condition is as follows:

$$\prod_{j=1}^{N_d} \sum_{i=1}^l (t_i = \bar{a}_{ij}) = 1$$

When the above expression evaluates to true, it means that none of the diagnoses were selected by conjunctions.

4.5.2 NOPAL Table Output

Table 4.10 shows the tabular representation of test modules for the CPS example. Each row in this table represents a test module. Columnwise the table is separated into two sections; (1) on the left side is the test module identification section, (2) on the right side is the diagnosis selection logic section. Under the test module identification section, the sequence entry gives the original sequence number of the assertion as it appears in the assertion limit and binary valued diagnosis table. The full description of the stimulus and measurement whose identification numbers are referenced can be found in the candidate tests library. The assertion whose designation number is given under the assertion entry can be found from the corresponding sequence number of the assertion limit table.

Under the diagnosis selection logic section, the table is divided into several columns. Each column is labelled by an equivalence class name. The failures which are in these classes are identified in the ambiguity report. These failures become the affected components (and failure functions) in the diagnosis message sent to the ATE operator.

The entries under their columns are the diagnosis selection logic operators (|, |&, &, &|). A diagnosis may be selected by the disjunction or conjunction of test modules. To simplify the representation they are shown in two

separate columns. The column on the left side contains the operators which select a diagnosis by disjunctions (i.e., independently of other test modules). The column on the right side contains the operators which select the same diagnosis with conjunctions (i.e., jointly with other test modules). If the outcome of an assertion of a test is true, then all of the diagnoses which have an "|" symbol in the disjunction column are selected. If the outcome is false, then the diagnoses which have "|¬" symbol are selected. After a diagnosis is selected, a message identifying the failures is sent to the operator right away. However, before a diagnosis can be selected by conjunctions ("&" or "&¬") all of the test modules which reference this diagnosis by conjunctions must be performed and the outcomes must accordingly select this diagnosis. The use of the negation symbol (¬) reverses the logical outcome of an assertion. If test module T_1 selects diagnosis D_1 by disjunction "|", then T_1 has to evaluate to true in order to select D_1 . Similarly, if T_1 selects diagnosis D_2 by negated disjunction "|¬", then T_1 has to evaluate to false to select diagnosis D_2 . The same type of negated logic is extended to selection by conjunctions. A blank entry in this section of the table means that the outcome of a test module for the corresponding diagnosis is not used (i.e., don't care).

It should be observed that each diagnosis may be selected by more than one test module with disjunctions. Throughout the complete testing only one diagnosis can be

TABLE 4.10 NOPAL Tabular Specification

| TEST MODULE IDENTIFICATION | | | | DIAGNOSIS SELECTION LOGIC | | | | | | | | |
|----------------------------|----------|-------------|-----------|---------------------------|---|---|---|---|---|---|---|---|
| SEQUENCE | STIMULUS | MEASUREMENT | ASSERTION | NOMINAL | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 5 | 3 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 6 | 3 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 | 3 | 2 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 8 | 3 | 2 | 4 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 3 | 2 | 5 | 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 10 | 3 | 2 | 6 | 6 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 11 | 3 | 2 | 7 | 7 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 12 | 3 | 2 | 8 | 8 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 13 | 3 | 2 | 9 | 9 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 14 | 3 | 2 | 10 | 10 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 22 | 5 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 23 | 5 | 4 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 24 | 5 | 4 | 3 | 3 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 25 | 5 | 4 | 4 | 4 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 26 | 5 | 4 | 5 | 5 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

selected by the conjunction of test modules. One of the minimum length test conjunctions does the selection. If none of the diagnoses can be selected by conjunctions, it means that there is an obvious error in testing. The most likely error is the existence of an unknown UUT failure. Another possibility is inaccurate modelling of the UUT in the circuit analysis.

If the testing is to be performed manually, this table can be used as a checklist to select the appropriate diagnoses.

4.5.3 NOPAL Specification Output

The final output of the FITS system is the NOPAL test specification (see Figure 4.6). This output is complete in the sense that it can be used as input to the bottom part of NOPAL without any modifications. The complete listing is not shown in Figure 4.6 because it is about 2300 lines long.

Before the NOPAL specification starts, the input files used and the output files generated are summarized. The name of a file is obtained from the user input. The date and time entries show when a file was created. Then comes the fault isolation summary. The organization of the summary table was explained in Section 4.3.6. The lines after the summary give statistical information about the test design. They include information relating to diagnosis percentage, longest and shortest test lengths and the average number of assertions used to select a diagnosis.

The statistics are self explanatory so they are not described further.

The first NOPAL statement is the specification identification. This name is the same as the UUT name. Each test module is identified by its sequence number. The stimuli, measurements and logic statements belong to the parent test module whose sequence number is given in parentheses. Stimuli and measurements also have unique identification numbers. They are numbered sequentially in the order they are listed. The stimulus or measurement identification appears just before the parent test module identification.

The first test module represents a resistance measurement test. An ohmmeter is connected between the UUT terminals J1_B and J1_E as a measurement device. The value of the resistance measured is assigned into the target variable ZJ1_B_2. A target variable name is constructed as follows. The first letter identifies the type of measurement. "Z" stands for resistance, "V" is for voltage, and "I" is for current measurements. Then comes the "from" UUT test terminal name where the measurement is taken (by notation the test terminals and circuit nodes are always written such that positive current flows from the first terminal to the second terminal). The measurement sequence number is suffixed to the terminal name to create a unique target variable name. This variable may be used by other test modules as a source variable.

After the target variable declaration are two assertion statements which belong to the measurement just described. These assertion statements evaluate to true or false depending on the measured value. The NOPAL processor then takes the logical and of the outcome of these two assertions. The logical outcome of the test module becomes the value of the logical and of the two assertions. The third assertion uses an IF-THEN-ELSE construct while declaring OUTCOME_1 as a target variable. Since this assertion declares a target variable, unlike the other two assertions, it is not used in determining the outcome of the test module. It merely stores the logical outcome of the test module. This variable is used at the end of testing to detect errors and to send appropriate messages. The name of the target variable is formed by suffixing "OUTCOME_" by the test module sequence number. This variable is undefined before the test is performed.

The logic statement gives the internal sequence number of the above assertion as found in the assertion limit and binary valued diagnosis table in a comment statement. If after the test is performed the logical outcome of the test module is true (measurement is within the limits), then diagnosis 29 is selected. Thus, the operator receives the message:

THIS IS AN INTERMEDIATE FAULT ISOLATION RESULT TESTING
CONTINUES ---
EC_SHORT(TQ2)

After this message is displayed, further processing determines that only test module 1 references diagnosis 63 by conjunction. So, the operator receives a second message:

FINAL FAULT ISOLATION RESULTS --- FAULTY COMPONENTS AND
THEIR FAILURE FUNCTIONS ARE LISTED AS
<FAILURE_FUNCTION>(<UUT_COMPONENT_NAME>):
EC_SHORT(TQ2)

At this point, the operator may terminate testing and print a repair ticket for the UUT or may continue to execute the other test modules to check for possible errors.

Test module 2 is an example where no stimulus is applied and no measurement is taken. It checks another range of values of a previously obtained measurement. The comment which appears immediately after the test module identification explains which test module actually performed the test whose result is being used. Then a dummy measurement identification is given. It is followed by some assertion statements and logic statements. They are similar to what happens in test module 1.

The last two test modules are used to send special messages to the operator. The conjunction of the assertions in test module 53 is true if diagnosis 35 was selected. This diagnosis is selected by the conjunction of several test modules to find out if the UUT is its NOMINAL state. If it is, then the operator gets the following message from diagnosis 72:

```
***** GOOD UUT *****  
(CHECK FAULT ISOLATION MESSAGE IF THERE ARE NO DIAGNOSIS  
FAILURES)
```

If there were any failures which could not be diagnosed, the operator would have received the proper list of failures. In this example there are a number of such failures. Therefore, the operator receives the following message, too:

```
THE FOLLOWING FAILURES (INCLUDING NOMINAL UUT) WERE NOT  
DIAGNOSED ---  
NOMINAL(ALL_COMPONENTS) | SHORT(R1) | SHORT(R2) |  
OPEN(R3) | SHORT(R6) | SHORT(R7) | OPEN(R10) | OPEN(TDCR1)  
| BC_SHORT(Q4) | OPEN(C1) | BE_SHORT(TQ3) | BE_SHORT(Q4) |  
SHORT(RSW)
```


If the outcome of the test module had been false, the following would have been sent:

```
***** BAD UUT *****  
  
TESTING PROCEEDS TO ISOLATE THE FAULTY COMPONENT
```

Then the faulty component would be isolated and reported to the operator as described in test module 1.

Test module 54 evaluates to true when none of the diagnoses can be selected by conjunctions of test modules. Then, the error message in diagnosis 74 is sent listing some of the probable causes of error that have taken place.

If the test module 54 evaluates to false, then only the fault isolation diagnosis message is sent.

After the descriptions of test modules are completed, a list of the failures of the UUT is given. This is the failure dictionary written in NOPAL syntax. Then all of the diagnoses and special messages are listed. This is followed by the UUT and ATE function declarations. Finally, the UUT test terminals are specified. The UUT_POINT is the external UUT test terminal name. The CONNECTOR name is the same as the test terminal input file name.

The last statement terminates the NOPAL specification for UUT_CPS.

The explanation of the NOPAL specification is completed here. As the reader may have already found, the comments in the NOPAL specification make it very easy to read through the code generated. If the reader is not content with the explanations given here, it is recommended that the formal definition given in the NOPAL user's guide be reviewed.

```

.....
/*
/*      NOPAL TEST SPECIFICATIONS GENERATED BY FITS VERSION 1.0
/*      AUTOMATED TEST DESIGN FOR ANALOG CIRCUITS
/*      DATE 01/01/77   TIME 12:00:00
/*
/*
.....
/*
/*      ACTIVE FILE IDENTIFICATION
/*
/*
=====
/* | FILE | NAME | DATE | TIME |
/* |-----|-----|-----|-----|
/* | UUT CIRCUIT DESCRIPTION | UUT_CPS | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/* | FAILURE DICTIONARY | DEFAULT | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/* | TEST TERMINALS | | | |
/* |-----|-----|-----|-----|
/* | INITIAL CONDITIONS | | | |
/* |-----|-----|-----|-----|
/* | OVERSTRESS | | | |
/* |-----|-----|-----|-----|
/* | ACCURACY SPECIFICATIONS | VERY-LOW | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/* | FAULT ISOLATION OBJECTIVES | | | |
/* |-----|-----|-----|-----|
/* | APPLICABLE TESTS LIBRARY - CANA | | | |
/* |-----|-----|-----|-----|
/* | APPLICABLE TESTS LIBRARY - ATE | | | |
/* |-----|-----|-----|-----|
/* | FAILURE SYMPTOM TABLE | UUT_CPS | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/* | TEST LIMIT AND DIAGNOSIS TABLE | ( 17) | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/* | MULTIPLE VALUED DIAGNOSIS TABLE | ( 17) | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/* | REDUCED MULT. VAL. DIAG TABLE | ( 8) | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/* | BINARY VALUED DIAGNOSIS TABLE | ( 99) | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/* | REDUCED BIN. VAL. DIAG TABLE | ( 52) | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/* | AMBIGUITY SUMMARY | | | |
/* |-----|-----|-----|-----|
/* | EQUIVALENCE CLASSES | | | |
/* |-----|-----|-----|-----|
/* | NUMBER OF FAILURE MODES | ( 67) | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/* | DIAGNOSIS PERCENTAGE | ( 50) | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/* | NUMBER OF EQUIVALENCE CLASSES | ( 34) | 01/01/77 | 12:00:00 |
/* |-----|-----|-----|-----|
/*
.....

```

Figure 4.16 NOPAL Specification Output

FAULT ISOLATION SUMMARY

DESIRED AND ACHIEVED LEVEL OF CUMULATIVE
FAULT ISOLATION PERCENTAGE

| K-AMBIGUITY | DESIRED C.F.I.P. | ACHIEVED C.F.I.P. | CLASS F.I.P. | NUMBER |
|-------------|---------------------|----------------------|-----------------|--------|
| 1 | 25.0% | 35.8% | 35.8% | 24 |
| 2 | 50.0% | 50.7% | 14.9% | 5 |
| 3 | 50.0% | 59.7% | 9.0% | 2 |
| 4 | 50.0% | 65.7% | 6.0% | 1 |
| 5 | 60.0% | 65.7% | 0.0% | 0 |
| 10 | 80.0% | 80.6% | 14.9% | 1 |

NUMBER OF FAILURE MODES : 67
NUMBER OF TESTS : 17
NUMBER OF EQUIV. CLASSES : 34
DESIRED LEVEL OF DIAGNOSIS: 80.0%
ACHIEVED LEVEL OF DIAGNOSIS: 80.6%
FAULT ISOLATION IS SATISFACTORY.

TOTAL NUMBER OF TEST SETUPS : 8
TOTAL NUMBER OF ASSERTIONS : 52
SHORTEST TEST SETUP : 1
LONGEST TEST SETUP : 8
SHORTEST ASSERTION CONJUNCTION : 1
LONGEST ASSERTION CONJUNCTION : 8
AVERAGE NUMBER OF ASSERTIONS/TEST : 6.50
AVERAGE NUMBER OF TEST/DIAGNOSIS : 1.44
AVERAGE NUMBER OF ASSERTION/DIAGNOSIS : 1.44


```

NOPAL SPECIFICATION UUT_CPS      ;

/*****
TEST      1 ;

    STIMULI      1( 1) ;
    /* THIS IS AN IMPEDENCE MEASUREMENT TEST - NO STIMULUS IS USED */

    MEASUREMENT  2( 1) ;
    CONJUNCTION ( 2) :
        ( < J1_B , J1_E > OHM = OHMMETER ( ZJ1_B_2 OHM ) )
        TARGET : ZJ1_B_2 ;

    ASSERTION : ZJ1_B_2 > 6.00000E03 ;          /* TEST LOWER LIMIT */
    ASSERTION : ZJ1_B_2 < 6.59000E03 ;          /* TEST UPPER LIMIT */

    ASSERTION      5( 2) :          /* SAVE THE OUTCOME OF THE TEST MODULE */
    IF ZJ1_B_2 > 6.00000E03 & ZJ1_B_2 < 6.59000E03
    THEN OUTCOME_1 = 1              /* OUTCOME OF THE TEST IS TRUE */
    ELSE OUTCOME_1 = 0              /* OUTCOME OF THE TEST IS FALSE */
    TARGET : OUTCOME_1 ;

    LOGIC ( 1) : /* INTERNAL SEQUENCE NUMBER ( 5, 3, 2, 1) */
        1 29 ;
        & 63 ;

*****/

TEST      2 ;
/* THERE IS NO NEED TO REPEAT PREVIOUSLY PERFORMED TEST
   THE VARIABLE USED IN THE FOLLOWING ASSERTIONS IS ALREADY
   AVAILABLE FROM STIMULUS ( 1) AND MEASUREMENT ( 2)
   OF TEST MODULE ( 1) */

    MEASUREMENT  3( 2) ;

    ASSERTION : ZJ1_B_2 > 6.74000E03 ;          /* TEST LOWER LIMIT */
    ASSERTION : ZJ1_B_2 < 6.76000E03 ;          /* TEST UPPER LIMIT */

    ASSERTION      6( 3) :          /* SAVE THE OUTCOME OF THE TEST MODULE */
    IF ZJ1_B_2 > 6.74000E03 & ZJ1_B_2 < 6.76000E03
    THEN OUTCOME_2 = 1              /* OUTCOME OF THE TEST IS TRUE */
    ELSE OUTCOME_2 = 0              /* OUTCOME OF THE TEST IS FALSE */
    TARGET : OUTCOME_2 ;

    LOGIC ( 2) : /* INTERNAL SEQUENCE NUMBER ( 6, 3, 2, 2) */
        1 32 ;
        & 66 ;

*****/

```

```

/*****
TEST 53 ;
/* FIND IF UUT IS OPERATIONAL OR MALFUNCTIONING
IN EITHER CASE INFORM THE OPERATOR
DIAGNOSIS ( 72) INDICATES EVERYTHING IS OK
DIAGNOSIS ( 73) INDICATES FAULT ISOLATION TESTS ARE ABOUT TO START */

MEASUREMENT 61( 53) ;

ASSERTION : OUTCOME_6 = 1 ;
ASSERTION : OUTCOME_15 = 1 ;
ASSERTION : OUTCOME_23 = 1 ;
ASSERTION : OUTCOME_30 = 1 ;
ASSERTION : OUTCOME_37 = 1 ;
ASSERTION : OUTCOME_43 = 1 ;
ASSERTION : OUTCOME_48 = 1 ;
ASSERTION : OUTCOME_50 = 1 ;

LOGIC ( 53) :
| 72 ; /* SEND UUT OPERATIONAL MESSAGE */
| 73 ; /* SEND UUT MALFUNCTIONING MESSAGE */

/*****

TEST 54 ;
/* IF NONE OF THE AFFECTED COMPONENT CLASSES CAN BE
SELECTED BY CONJUNCTIONS OF TEST MODULES THERE IS
A SIGNIFICANT ERROR IN TEST DESIGN -- POSSIBLE
REASONS OF ERROR ARE LISTED IN THE OPERATOR MESSAGE */

MEASUREMENT 62( 54) ;

ASSERTION : /* AND IF DIAGNOSIS ( 35) WAS NOT SELECTED */
IF OUTCOME_6 = 0 | OUTCOME_15 = 0 | OUTCOME_23 = 0 |
OUTCOME_30 = 0 | OUTCOME_37 = 0 | OUTCOME_43 = 0 |
OUTCOME_48 = 0 | OUTCOME_50 = 0
THEN 1=1 /* TRUE, DIAGNOSIS WAS NOT SELECTED */
ELSE 1=0 ; /* FALSE, DIAGNOSIS WAS SELECTED */

ASSERTION : /* AND IF DIAGNOSIS ( 36) WAS NOT SELECTED */
IF OUTCOME_6 = 0 | OUTCOME_15 = 0 | OUTCOME_27 = 0 |
OUTCOME_44 = 0
THEN 1=1 /* TRUE, DIAGNOSIS WAS NOT SELECTED */
ELSE 1=0 ; /* FALSE, DIAGNOSIS WAS SELECTED */

.
.
.
ASSERTION : /* AND IF DIAGNOSIS ( 40) WAS NOT SELECTED */
OUTCOME_29 = 0 ;

.
.
.

LOGIC ( 54) : /* SEND MAJOR ERROR MESSAGE */
| 74 ;

/*****

```

```

/...../
/*                                          */
/*          UUT COMPONENT FAILURE DICTIONARY          */
/*                                          */
/...../

```

| | | | | |
|-----------|----|-------------|----------------------|--------------|
| COMP_FAIL | 1 | : ALL_COMPS | , FUNCTION=NOMINAL | , INDEX= 0 ; |
| COMP_FAIL | 2 | : R1 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 3 | : R1 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 4 | : R2 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 5 | : R2 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 6 | : R3 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 7 | : R3 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 8 | : R4 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 9 | : R4 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 10 | : R5 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 11 | : R5 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 12 | : R6 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 13 | : R6 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 14 | : R7 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 15 | : R7 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 16 | : R8 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 17 | : R9 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 18 | : R9 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 19 | : R10 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 20 | : R10 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 21 | : R11 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 22 | : R11 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 23 | : R10 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 24 | : R10 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 25 | : C1 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 26 | : TDCR1 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 27 | : TDCR1 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 28 | : TDCR2 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 29 | : TDCR3 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 30 | : TDCR4 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 31 | : TDCR4 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 32 | : TDCR5 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 33 | : TDCR5 | , FUNCTION=SHORT | , INDEX= 0 ; |
| COMP_FAIL | 34 | : T1A | , FUNCTION=COLL_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 35 | : T1A | , FUNCTION=BASE_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 36 | : T1A | , FUNCTION=EMIT_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 37 | : T1A | , FUNCTION=BC_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 38 | : T1A | , FUNCTION=RE_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 39 | : T2A | , FUNCTION=EC_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 40 | : T2B | , FUNCTION=COLL_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 41 | : T2B | , FUNCTION=EMIT_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 42 | : T2B | , FUNCTION=BC_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 43 | : T2B | , FUNCTION=HE_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 44 | : T2B | , FUNCTION=EC_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 45 | : T2 | , FUNCTION=BASE_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 46 | : T2 | , FUNCTION=EMIT_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 47 | : T2 | , FUNCTION=BC_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 48 | : T2 | , FUNCTION=UE_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 49 | : T2 | , FUNCTION=EC_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 50 | : T3 | , FUNCTION=COLL_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 51 | : T3 | , FUNCTION=BASE_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 52 | : T3 | , FUNCTION=EMIT_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 53 | : T3 | , FUNCTION=BC_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 54 | : T3 | , FUNCTION=EC_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 55 | : T4 | , FUNCTION=BASE_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 56 | : T4 | , FUNCTION=EMIT_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 57 | : T4 | , FUNCTION=BC_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 58 | : T4 | , FUNCTION=EC_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 59 | : C1 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 60 | : TDCR2 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 61 | : TDCR3 | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 62 | : T2 | , FUNCTION=COLL_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 63 | : T3 | , FUNCTION=HE_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 64 | : T4 | , FUNCTION=COLL_OPEN | , INDEX= 0 ; |
| COMP_FAIL | 65 | : T4 | , FUNCTION=BE_SHORT | , INDEX= 0 ; |
| COMP_FAIL | 66 | : RSW | , FUNCTION=OPEN | , INDEX= 0 ; |
| COMP_FAIL | 67 | : RSW | , FUNCTION=SHORT | , INDEX= 0 ; |

```

/...../

```



```

...../
/*
/*          DIAGNOSES AND SPECIAL MESSAGES          */
/*
...../

```

```

DIAGNOSIS 1 :
  OPERATOR MESSAGE :
    AFFECTED COMPONENTS =
      NOMINAL(ALL_COMPS) | SHORT(R1) | SHORT(R2) |
      OPEN(R3) | SHORT(R6) | SHORT(R7) |
      OPEN(R10) | OPEN(TDCR1) | BC_SHORT(TQ4) |
      OPEN(C1) | BE_SHORT(TQ3) | BE_SHORT(TQ4) |
      SHORT(RSW) ,
    TYPE= DISJUNCTION ;

```

```

.
.
.

```

```

DIAGNOSIS 29 :
  OPERATOR MESSAGE :
    AFFECTED COMPONENTS =
      EC_SHORT(TQ2) ,
    TYPE= DISJUNCTION ;

```

```

.
.
.

```

```

DIAGNOSIS 35 :
  OPERATOR MESSAGE :
    AFFECTED COMPONENTS =
      NOMINAL(ALL_COMPS) | SHORT(R1) | SHORT(R2) |
      OPEN(R3) | SHORT(R6) | SHORT(R7) |
      OPEN(R10) | OPEN(TDCR1) | BC_SHORT(TQ4) |
      OPEN(C1) | BE_SHORT(TQ3) | BE_SHORT(TQ4) |
      SHORT(RSW) ,
    TYPE= NODIAGNOSIS ;

```

```

.
.
.

```

```

DIAGNOSIS 63 :
  OPERATOR MESSAGE :
    AFFECTED COMPONENTS =
      EC_SHORT(TQ2) ,
    TYPE= CONJUNCTION ;

```

```

.
.
.

```

```

...../

```

```

/...../
MESSAGE DISJUNCTION :
TEXT="THIS IS AN INTERMEDIATE FAULT ISOLATION RESULT
TESTING CONTINUES ---- ( C ) " ;

MESSAGE CONJUNCTION :
TEXT="FINAL FAULT ISOLATION RESULTS ---
FAULTY COMPONENTS AND THEIR FAILURE FUNCTIONS
ARE LISTED AS <FAILURE_FUNCTION> ( <UUT_COMPONENT_NAME> ) :
( C ) " ;

MESSAGE NODIAGNOSIS :
TEXT="THE FOLLOWING FAILURES (INCLUDING NOMINAL UUT)
WERE NOT DIAGNOSABLE -- ( C ) " ;

/...../

DIAGNOSIS 72:
OPERATOR MESSAGE :
TYPE= GOOD_UUT :
MESSAGE GOOD_UUT :
TEXT="**** GOOD UUT ****
(CHECK FAULT ISOLATION MESSAGE TO SEE IF THERE ARE
NO-DIAGNOSIS FAILURES) " ;

DIAGNOSIS 73:
OPERATOR MESSAGE :
TYPE= BAD_UUT :
MESSAGE BAD_UUT :
TEXT="**** BAD UUT ****
TESTING PROCEEDS TO ISOLATE THE FAULTY COMPONENT" ;

DIAGNOSIS 74:
OPERATOR MESSAGE :
TYPE= MAJOR_ERROR :
MESSAGE MAJOR_ERROR :
TEXT="**** MAJOR ERROR IN FITS FAULT ISOLATION LOGIC ****
PROBABLE CAUSES OF ERROR ARE :
(1) UNDEFINED FAILURE OF UUT,
(2) INACCURATE MODELLING IN CIRCUIT ANALYSIS,
(3) INSUFFICIENT ATE ACCURACY,
(4) ATE OPERATOR ERROR IN MANUAL TEST SETUP (IF ANY). " ;

/...../

```

```

/*****
/*
/*          UUT AND ATE FUNCTIONS
/*
*****/

FUNCTION : NOMINAL      , FUNCTION TYPE=FAILURE, PARAM=(COMP,S) ;
FUNCTION : OPEN         , FUNCTION TYPE=FAILURE, PARAM=(COMP,S) ;
FUNCTION : SHORT        , FUNCTION TYPE=FAILURE, PARAM=(COMP,S) ;
FUNCTION : COLL_OPEN    , FUNCTION TYPE=FAILURE, PARAM=(COMP,S) ;
FUNCTION : BASE_OPEN    , FUNCTION TYPE=FAILURE, PARAM=(COMP,S) ;
FUNCTION : EMIT_OPEN    , FUNCTION TYPE=FAILURE, PARAM=(COMP,S) ;
FUNCTION : BC_SHORT     , FUNCTION TYPE=FAILURE, PARAM=(COMP,S) ;
FUNCTION : BE_SHORT     , FUNCTION TYPE=FAILURE, PARAM=(COMP,S) ;
FUNCTION : EC_SHORT     , FUNCTION TYPE=FAILURE, PARAM=(COMP,S) ;

FUNCTION : VOLTMETER, TYPE=M, #PINS=2,
          PARAM1=(V,T,(VOLT,+50,-50))
          PARAM2=(R,S,(OHM,10MEG,10MEG)) ;
FUNCTION : AMPMETER, TYPE=M, #PINS=2,
          PARAM1=(I,T,(AMP,+1.0,-1.0))
          PARAM2=(G,S,(MHO,100,100)) ;
FUNCTION : OHMMETER, TYPE=M, #PINS=2,
          PARAM1=(R,T,(OHM,10MEG,0.1))
FUNCTION : ESUPPLY, TYPE=S, #PINS=2,
          PARAM1=(V,S,(VOLT,50,-50))
          PARAM2=(R,S,(OHM,1E-6,1E-6)) ;
FUNCTION : JSUPPLY, TYPE=S, #PINS=2,
          PARAM1=(I,S,(AMP,1.0E-6,-1.0E-6))
          PARAM2=(G,S,(MHO,1E-6,1E-6)) ;

/*****
/*
/*          UUT TEST TERMINALS
/*
*****/

UUT_POINT : J1_B      , CONNECTOR=( J1_PIN      ) ;
/* GROUND
UUT_POINT : J1_A      , CONNECTOR=( J1_PIN      ) ;
/* Q_SWITCH
UUT_POINT : J1_E      , CONNECTOR=( J1_PIN      ) ;
/* +24V_AUX.
UUT_POINT : J1_F      , CONNECTOR=( J1_PIN      ) ;
/* +15V
UUT_POINT : J1_G      , CONNECTOR=( J1_PIN      ) ;
/* +25V
UUT_POINT : J1_H      , CONNECTOR=( J1_PIN      ) ;
/* +28,+24 TANK BATTERY SENSE
UUT_POINT : J1_X      , CONNECTOR=( J1_PIN      ) ;
/* BT1_RECHARGABLE BATTERY

/*****
/*
/*          END OF NOPAL TEST SPECIFICATIONS
/*
*****/

END UUT_CPS      ;

```


CHAPTER 5

DESIGN OF THE FITS SYSTEM

In this chapter some of the design considerations and problems associated with fault detection and isolation are discussed. Section 5.1 describes the types of circuits which can be analyzed with the FITS methodology. Section 5.2 contains a discussion of the failures which are observed in analog circuits. More prominent catastrophic failures of basic electronic components are emphasized. Section 5.3 describes the interpretation of test setup. Section 5.4 lists the assumptions made in this work. Section 5.5 contains a discussion of the methods used to find test limits using fault simulation. In Section 5.5 a formalization of the fault isolation logic used in the FITS system is described. Finally in Section 5.6, the test minimization method is described.

5.1 Types of Circuits to be Tested

One of the objectives in the design of the FITS system is to make it possible to generate tests to diagnose and isolate failures for a large variety of electronic analog circuits. It is not possible to include all types of analog circuits. The class of analog circuits which are preferred have an observable steady state. That is, test specifications for circuits which exhibit a constant level of measurable output response at their available test terminals when attached to fixed power sources can be generated automatically. This restriction excludes devices such as oscillators and function generators as immediate candidates for automatically testable circuits. However, through appropriate test strategies defined by the user, test design for such devices can also be accomplished by FITS.

The reason why digital circuits are excluded from investigation is essentially due to the technology advancement in digital circuit design. Digital circuits are no longer constructed using the basic circuit elements such as resistors and transistors, but they are constructed by using functional units such as logic gates, counters and shift registers. A single circuit component may include hundreds of basic analog circuit elements in one package, but it generally has only a few input-output terminals and power supply terminals. For instance, microprocessors in the MCS-48 series have about 20000 transistors in one IC package

[83]. Digital components are used to perform logic functions, and their internal construction is irrelevant to the interest of the designer. Therefore, the troubleshooting of such devices is performed at a functional level, and components are replaced at the IC package level. The lowest level in digital circuit testing may be structural to assure that each individual hardware constituent operates correctly. However, it is much faster to perform the testing at a functional level. In functional test design, a list of input and output response patterns are found to diagnose and isolate the failures. The most common approach assumes stuck-at-one or stuck-at-zero (SA Model) failures of digital devices. Then, test string patterns are found using methods like path-sensitization [84] or Boolean difference [85,86] of the Boolean expressions which describe the system behavior.

Because of the emphasis put on functional testing analog circuit testing is differentiated from digital testing. However, occasionally both analog and digital testing use the similar terminology and fault diagnosis and isolation techniques. After all, digital circuits are essentially analog circuits when they are investigated at the basic circuit element level.

The FITS system enforces no restrictions on the size, function or interfaces required for the proper operation of the UUT. It may have both linear and nonlinear characteristics. If the user wishes, the environmental

dependencies such as radiation and temperature may be included. The circuit is described to the FITS system as if it is being described for analysis by a CANA program. The power supplies may be specified as the initial conditions to find the nominal quiescent operating conditions. If the circuit has more than one nominal operating mode (involving different states of thermal switches, mechanical switches, push-buttons, etc), each one of these conditions needs to be described as a different initial state of the circuit. If there are memory states, these states need to be described as separate initial states.

FITS is primarily intended for diagnosing analog circuits which use only discrete components. If the circuit includes complex devices such as operational amplifiers, they need to be modelled in terms of basic circuit components or as functional convolutions. This imposes no restrictions on the circuit types, since such devices must be described to a circuit analysis program in this manner anyway.

The nominal circuit is expected to be described in terms of the nominal component values with their tolerances as specified in the manufacturer's specification lists. The component tolerances may be assigned wider tolerances to accommodate less stringent performance as desired by the designer (see Section 5.5).

5.2 Types of Failures

The malfunction of a UUT due to changes in component properties is a failure by definition. The state the UUT is in when it has a failure is called a failure mode. In this work, the failures which are of primary concern are single catastrophic failures of individual circuit components. Composite failures which are defined by the failures of several components are less likely to happen than single catastrophic failures, so composite failures are not emphasized in this work. Open or short resistors, capacitors, inductors and diodes are generally considered to be catastrophic failures. Similarly, open or short terminals of other semiconductor devices are catastrophic failures. For example, shorted collector-emitter junction of bipolar transistors is a common failure. Since any circuit response may be a symptom of a failure by definition, any component value may be the reason of a failure. Thus, 20% degradation of the dc-gain characteristic of a transistor may be defined to be a failure. These failures are modelled as component value changes or topological changes in the circuit diagram. A failure is assumed to be stable (fixed) throughout the testing time.

The FITS system presumes that all failure modes which the UUT may have are well defined and adequately described to the system in the failure dictionary. Each failure mode exhibits a different state of the UUT operation. The

nominal operating mode is treated just as another state of the UUT. By definition the nominal mode is the failure number 1 (F_1).

Even though the system was developed for detecting and isolating single failures, it may readily be extended to include multiple or induced failure modes of the UUT. This is done by describing these composite failures as changes to the nominal circuit and giving an identification name to the failure mode.

All failure modes are assumed to be equally likely to occur. More prominent failures may optionally be assigned a relative frequency index which is passed to the bottom part of NOPAL as a parameter to influence the execution sequence of the tests generated.

The basic failure functions of electronic components are unified under a general theory [87]. It is necessary to develop the failure functions of components from published data or after performing experiments to determine the failure mechanisms in relation to the area of interest.

A large amount of information is available from general reliability analysis to define and calculate the failure rate indices for component failures. Failure rates of elements and the mean—time—between—failures (MTBF) are useful concepts in determining what to include in the failure dictionary. Certain types of failures, such as a resistor failing by shorting or capacitor by open circuit, are unlikely to happen. They are evidenced from the data

available from reliability analysis. Similarly, it is observed that the failure rates of certain components are higher than others. For example, transistors fail more frequently than resistors.

The books "Probabilistic Reliability" by Shooman [87] and "Fundamentals of Reliability Theory" by Polovko [88] contain several chapters relating to component failures and reliability calculations. They both contain interpretations of the failure rate data available in "Reliability Stress and Failure Rate Data" (MIL-HDBK-217, 217A and 217B) [89] and "Failure Rate Data Book - FARADA" [90]. They include information on the total number of tests, number of failures, source of the data and the environment. It is recommended that users of the FITS system who are intending to use the failure definition option familiarize themselves with the concepts presented in these references.

A short summary of the failure functions found in analog circuits is presented here for uniformity. Generally, failures of circuits are due either to chance failures or to wear-out failures of individual components. The wear-out failures are caused by gradual changes in component characteristic with time, such as drift in value of resistance. The chance failures are observed as catastrophic failures such as open or short circuits. The wear-out failures are observed as degradation of performance.

The chance failures are characterized by reliability R

which is exponentially related to the failure rate λ and to the operating time t ($0 < t < +\infty$) by:

$$R(t) = \lambda e^{-\lambda t}$$

The wear-out failures are characterized by the drift of parameters beyond admissible limits. The wear-out failures follow the normal distribution law:

$$R(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(t-\mu)^2/2\sigma^2}$$

where $-\infty < t < +\infty$, σ is standard deviation and μ is a constant.

Statistical data indicate that nearly 50% of the total number of components in analog circuits (whether they are transistor dominant, discrete device or otherwise) are resistors [91]. Even though resistors are among the most reliable components, due to the large number of occurrences a circuit may fail due to failures in resistors. More than 55% of the failures in resistors are due to various open circuits, and 35-40% are specifically due to the burn-out of the conducting material. Hence 90-95% of the failures result in open circuits. Only about 5% of the failures are due to sharp decrease of resistance (shorting) [88]. Resistor failures may further be identified for semiconductor, carbon, wire-wound or composition type materials and according to MTBF depending on the loading factor, ambient temperature, humidity and the frequency of the applied voltage. These details are not further described for they are beyond the scope of this discussion.

About 27-33% of the components in transistor-IC and discrete device circuits are capacitors [91]. Capacitors are also among the reliable components. About 80% of capacitor failures are due to the puncture of the dielectric and flash-over (surface discharge) in the insulation between plates. The failures which are due to a nonpermissible decrease of insulation resistance amount to about 5% of all failures. Open capacitor failures amount to about 15% of the total failures [88]. As for resistors, the MTBF of capacitors depends on the loading factor, ambient temperature, humidity and the frequency of the applied voltage.

Transformers, chokes and coils may fail in modes of: open circuit in the winding, interwinding shorts, puncture to the case, and change of fundamental parameters.

Relays and switches are among the least reliable components because of the mechanical motion involved. They fail by the sticking of contacts and by failures similar to inductive elements.

A diode may fail in two modes; open or short circuit. When shorted it behaves as a very low resistance, and when open it behaves as infinite resistance in both directions.

A bipolar transistor may operate in three regions - saturation, normal and cutoff regions. There may be different modes of failure in each region. If the transistor never operates in that region, the failure may not effect circuit performance at all. It has been reported

that bipolar transistors may have about 30 possible failures due to almost 200 failure mechanisms [92]. Common types of failures are shorts between the junctions and open of junction terminals [93]. High level of noise generation is also a common failure.

5.3 Test Setup

In this work a test setup refers to the configuration of connecting test devices to a UUT. Test devices of a test setup may be composed of one or more stimulus devices and a measurement device connected to the UUT at its available test terminals (see Figure 5.1). A stimulus device is effectively a function which influences the UUT in any fashion. These devices may be power supplies, loads, heating or cooling mechanisms, etc. A measurement device is effectively a function which quantifies the response of the UUT to a stimulus. It may be a physical quantity or a derived quantity. For example, the measurement of the gain of an amplifier is performed by calculating the ratio of the output power to the input power. A measurement taken by a measurement device is a single number. The measurement of a UUT response due to a stimulus at given test terminals is simply called a test. When a test is named, the stimulus and measurement devices as well as their connection terminals are specified. When a measurement name is referenced, the stimulus associated with the measurement can be found in the test identification.

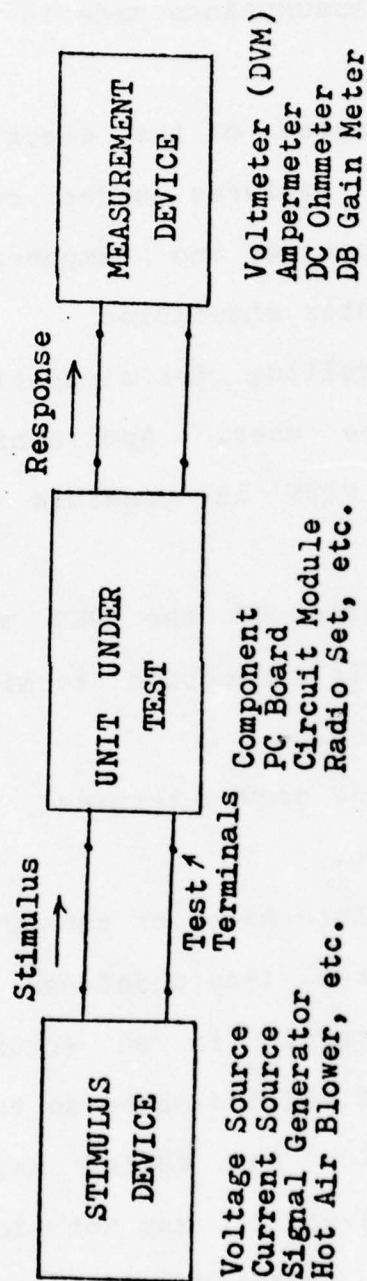


Figure 5.1 Test Setup

5.4 Assumptions

A list of the assumptions made in this work are given below.

1. The circuit diagram of the electronic analog circuit (UUT) and the failures under consideration can be described by topology and component values to a CANA system for computer simulation.
2. Accuracy in modelling for simulation is provided and verified by the user. Applicability of the tests generated by FITS is possible only through this provision.
3. All power supplies of the UUT are external to the circuit and, their connection terminals are accessible for ATE interfaces.
4. In addition to the ground terminal, the UUT has at least one test terminal.
5. All possible failure modes of the UUT are defined in the failure dictionary. (Any undefined failure mode may go undetected, or result in an erroneous diagnosis, or be properly detected and indicated so to the user.)
6. A component which has failed may overstress other components; however, it may not induce an additional failure in the overstressed components unless the composite failure mode is stated explicitly as a distinct failure mode in the dictionary.
7. CANA simulation and physical testing are compatible. That is, the simulated circuit response can be measured

by the ATE within the specified measurement tolerance. The ATE should be capable of applying the stimuli and taking the measurements as specified. The environmental conditions such as temperature and humidity levels in the ATE environment should be duplicated as specified to the simulation program.

8. The UUT does not have an oscillatory response when it has one of the possible failure modes listed in the failure dictionary.
9. The failure of a UUT is fixed; that is, the UUT with a failure does not change its properties while the testing is in progress. The UUT response and the definition of the failure are time invariant.

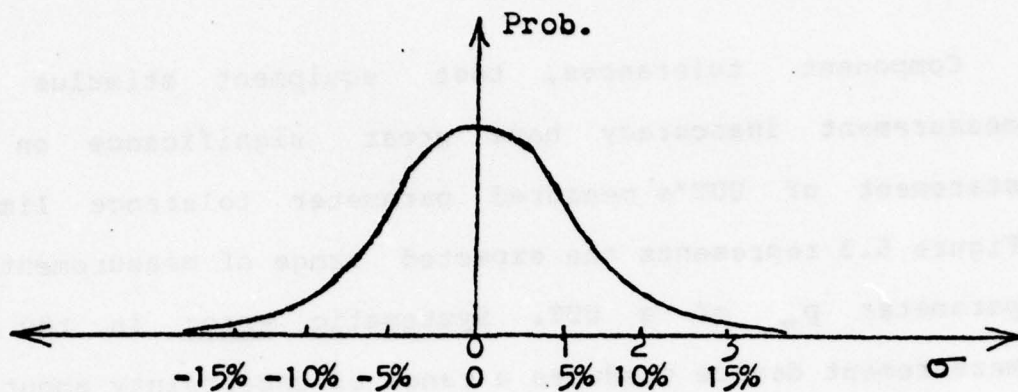
5.5 Measurements to Detect Symptoms of Failure

All measurable quantities have uncertainties. Because of the varying tolerances of the components used in building electronic circuits, no two units have identical properties. Even if they have, the measurements made on them may be slightly different because of the measurement device inaccuracy. Tolerance analysis is an involved research area. A discussion of related topics can be found in "Computer Aided Network Design" by Calahan [94] and in the April 1971 issue of the Bell System Technical Journal [95].

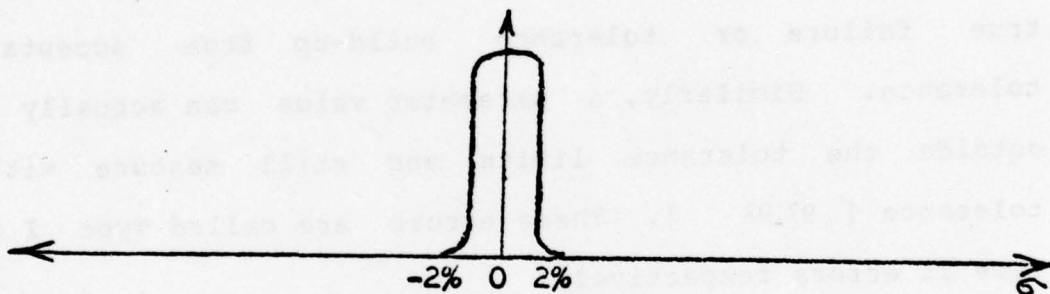
During the design stage of circuits, great attention is paid to restrict the tolerances on the components such that any unit built using off-the-shelf components will

perform within specifications. Usually there are three types of tolerance distributions of the electronic components available from manufacturers. They are characterized by normal, bimodal and uniform distributions as illustrated in Figure 5.2. The components with normal distribution of tolerance are most widely used in commercial applications. If the components have narrow tolerance limit, the distribution of their values approaches uniform distribution over the tolerance band. If the uniformly distributed components are removed from a normal distribution, the remainder exhibits a bimodal distribution. Uniform distribution is generally used in high precision equipment and bimodal distribution is used in lower quality products.

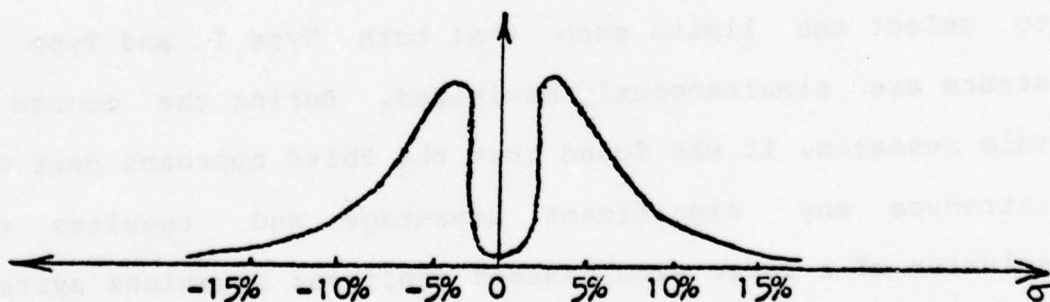
The effect of using components with values different from the nominal results in UUT response which varies over a range of values for different UUTs. The central limit theorem [96] shows that when there are a large number of independent variables, normal distribution is approached even when each variable is not normally distributed. Hence the net effect of independent random variations of component values may be assumed to be normally distributed for all measurements made at the test terminals of a UUT. This assumption may not always be valid, however it is observed that the majority of electronic circuits are characterized by this property.



NORMAL DISTRIBUTION
 ($3\sigma = \pm 15\%$ Manufacturing Tolerance)



UNIFORM DISTRIBUTION
 (Distribution of Selected Component Values $\pm 2\%$)



BIMODAL DISTRIBUTION
 (The Selection Process Produces Bimodal Distribution of the Remaining Components)

Figure 5.2 Component Tolerance Distributions

5.5.1 Test Limits Specification

Component tolerances, test equipment stimulus and measurement inaccuracy have great significance on the statement of UUT's measured parameter tolerance limits. Figure 5.3 represents the expected range of measurements of parameter p_m of a UUT. Systematic error in the ATE measurement device produces a range of uncertainty about any measured value. Therefore, a measurement which is slightly outside the indicated tolerance limits for a particular parameter measurement instance could possibly represent a true failure or tolerance build-up from acceptable tolerance. Similarly, a parameter value can actually be outside the tolerance limits and still measure within tolerance [97,98]. These errors are called Type I and Type II errors, respectively.

Test limits can be adjusted at either extreme so that no acceptable performance is rejected, or no marginally out-of-tolerance performance is accepted. A third choice is to select the limits such that both Type I and Type II errors are simultaneously minimized. During the course of this research, it was found that the third approach does not introduce any significant advantage and requires the solution of a large simultaneous nonlinear equations system. Minimization of only Type I errors is widely practiced. So, this approach is adopted in the FITS system (Figure 5.4). The system can readily be modified to minimize Type II errors.

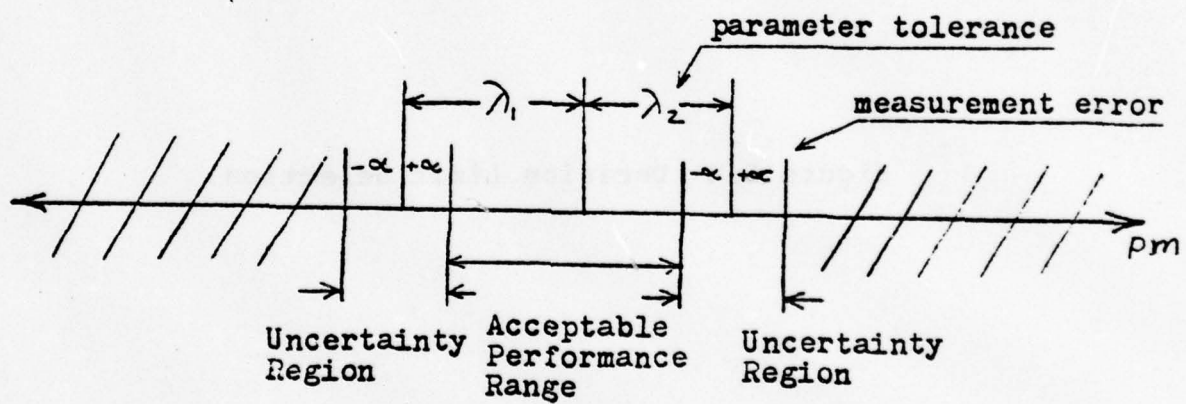
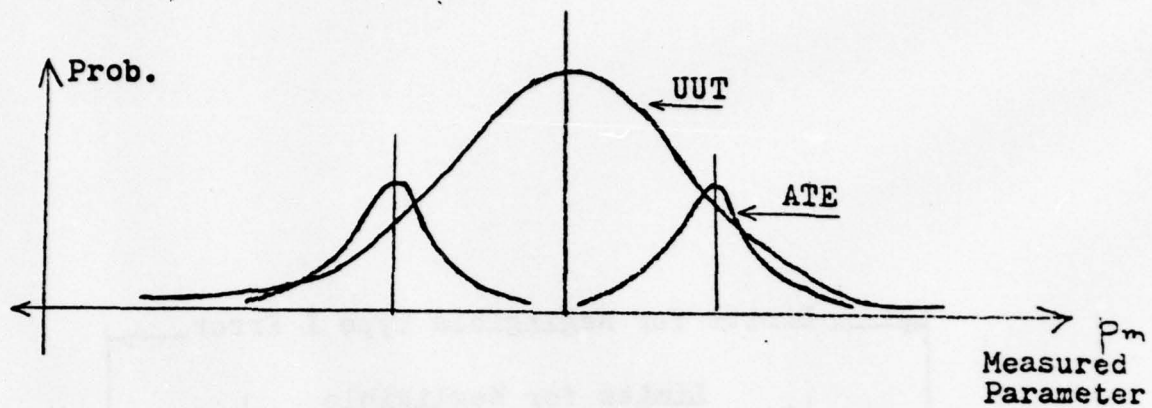


Figure 5.3 Parameter Tolerance Limits

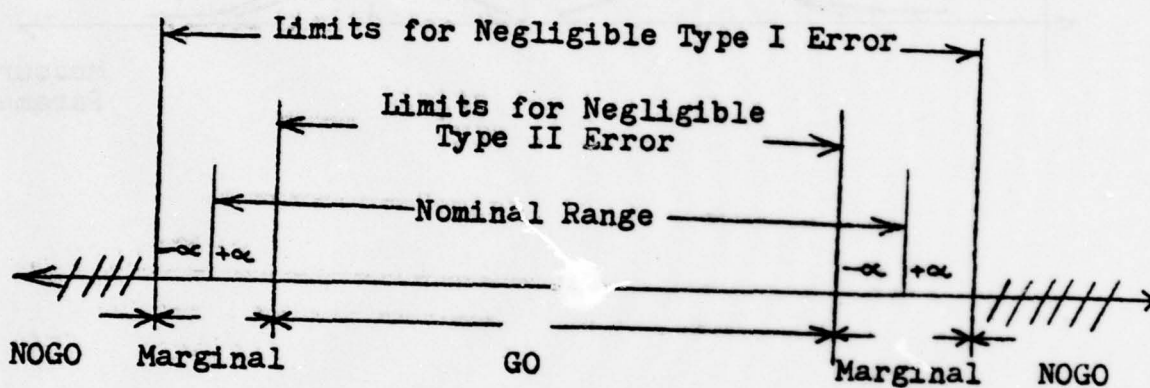


Figure 5.4 Decision Limit Selection

5.5.2 Calculation of Parameter Limits

In order to determine the test limits for symptoms of failure, the following four techniques can be employed: (1) empirical tests, (2) statistical analysis, (3) worst-case analysis, and (4) Monte Carlo techniques.

1. Empirical Tests: This technique is the most difficult and time consuming to perform. When a circuit is built using breadboard techniques, its specifications result in requiring precision components so that production units match the breadboard prototype. Therefore, when this design technique is used, the diagnostic test limits should not be based on the component tolerance specifications. If the test limits are based on them, they will be much tighter than necessary for acceptable performance. Establishing test limits in this case is extremely difficult. The most common practice is to insert the prominent catastrophic failures to observe the symptoms of failure.

The major drawback of this approach is its vulnerability to test design error and the requirement of specially skilled test technicians to conduct the tests. In the FITS system such tests are recommended for test design verification instead of test design.

2. Statistical Analysis by Sensitivity Approach: When the tolerances of the individual components are statistically independent, tolerances of components may

be added in root-sum-squared manner to yield the overall tolerance. Root-sum-square (RSS) method enables to obtain a statistical estimate of the deviation of the parameter (p_i) to be measured by computing the square root of the sum of the squares of the sensitivities weighted by the tolerance interval of each component for which a tolerance is specified. The sensitivity S_{ij} of parameter p with respect to component C_j is defined as a partial derivative:

$$S_{ij} = \frac{\partial p_i}{\partial C_j} C_j$$

The RSS of p_i is obtained by:

$$RSS_{p_i} = \sqrt{\sum_{j=1}^n (S_{ij} \Delta tol_j)^2}$$

where Δtol_j is the relative tolerance interval of component C_j .

The RSS value is closely related to the standard deviation of the measured parameter. If the number of randomly varying circuit components is large, the distribution of the measured response approaches the normal distribution. Thus the circuit response can be expected to be measured within a few RSS values away from the nominal.

This technique is generally used with a CANA program to determine the sensitivity of the parameter. The root-sum-square sensitivities and component

tolerances are used to predict the range of response. This approach does not give a true indication of the parameter ranges since the sensitivity by definition is a local behavior. Statistical tolerance assignment is generally used in commercial environment where component part cost is a significant design factor. In military systems, due to the performance requirements and environmental considerations, component parts have tighter-than-necessary tolerance specifications for circuit performance. Therefore, it may be misleading to base test limit selection criteria solely on component tolerance specification in the purchase orders. FITS provides the RSS values of all measurements calculated to aid the user in observing the effects of component tolerances.

Detailed description of this type of analysis can be found in several references [94,99, 100, 101]. PCAP, SPICE2, ISPICE, IMAG-III, EXTENDED SCEPTRE and NAP2 circuit analysis programs are capable of performing sensitivity analysis.

3. Monte Carlo Techniques: This is essentially a statistical sampling technique. Component part values are randomly selected between allowable tolerance bounds, and test measurements are calculated. The sample mean and standard deviation are taken as approximations to the expected measurement limits [94,

102]. A large number ($n > 30$) of samples need to be analyzed to obtain reliable statistics. This technique is invariably applied with CANA programs. ASTAP, BELAC and EXTENDED SCEPTRE are capable of performing Monte Carlo analysis.

4. Worst-Case Analysis: This technique provides the simplest method to calculate parameter tolerances using CANA programs. Two approaches are possible: (1) true worst-case analysis, and (2) approximate worst-case analysis. In the true worst-case analysis, component values are varied within tolerance by small increments to detect the absolute minimum and maximum values of the measured parameter. Approximate worst-case analysis makes use of the direction of the sensitivity at nominal measurement value to perturb the component values to their tolerance limits. The sensitivity is calculated at either extreme to detect any sensitivity sign (slope) change which would be an indication that the calculated worst-case is not the true worst-case (i.e., the true worst-case lies somewhere between the tolerance limits). Even when the signs don't change, there is no assurance that the calculated value is the true minimum or maximum since the sign may have changed more than once in the interval [94,103,104]. BELAC, NAP2, EXTENDED SCEPTRE and UCCAP are capable of performing worst-case analysis.

Any one of the above techniques (except the empirical test) can be automated using a computer to determine the limits on measurements. Worst-case analysis is incorporated into the FITS system because of the programming convenience and the speed in computation. In most cases the worst-case computation takes only a few (2) more iterations after convergence to the nominal solution. The fast computation of network parameter sensitivities and approximate worst-case are made possible by using the adjoint network of the original network due to Tellegen's theorem [105]. A significant portion of the solution time is spent to calculate the nominal solution which takes well over 5 iterations (about 10 iterations for most circuits with initial conditions provided) [106]. Approximate worst-case calculation does not introduce a significant overhead to determine the test limits.

5.6 Abstraction of Fault Isolation Logic

The principal idea behind the fault isolation method can be simply expressed as the "classification" of failures. In this work it is assumed that the failures of the UUT are predetermined. That is, before test design is initiated the failures which the UUT may have are known. The intention is to find characteristic symptoms of the failures so that the following questions can be answered: (1) Does the UUT have any one of these predetermined failures? (2) If it has one, how well can this particular failure be identified? The question of what happens when the UUT has an unknown failure

is not directly addressed. But there is provision to detect some of these undefined failures. It is hard to predict what the diagnosis will be when such a failure is encountered. Two possible outcomes of such cases are: (1) the tests will erroneously select a diagnosis and isolate the unknown failure as one of the predetermined failures or (2) none of the diagnoses of the tests (by conjunctions) will be selected, indicating that there is an unknown failure in the UUT.

The measurements made on a UUT can be used to identify the coordinates ($M_i : i=1,2, \dots, n$) of an n -dimensional failure symptoms space (S-space). All failure modes ($F_j : j=1,2,\dots,N_F$) of the UUT result in a measurement range (failure symptom) along the measurement coordinates. The subspaces bound by the specification of failure symptoms on each coordinate define the S-cubes (see Figure 5.5). Some of these ranges overlap. The lower and upper limits of the nonoverlapping ranges are called assertion limits. The assertion limits along each coordinate is used to define regions in the S-space. The space bound the assertion limits along the n coordinates define n -dimensional hypercubes (A-cubes) in the failure symptoms space. The total number of possible A-cubes is $N_{total} = \prod_{i=1}^n n_{M_i}$ where n_{M_i} is the number of assertions of measurement M_i . All of the S-cubes are contained in the larger A-cubes. Not all of the A-cubes of this space are allowed. If an A-cube contains a valid S-cube, it defines an allowed cube A-cube. All regions

outside the allowed A-cubes constitute forbidden measurement regions. The highest number of allowable hypercubes is therefore equal to the number of failure modes N_F . In most cases, the number of allowable A-cubes is less than N_F because some failure symptoms overlap on the measurement coordinates. The A-cubes which do not contain any S-cube should not be measurable at all. Such A-cubes and all the space outside the remaining A-cubes (forbidden regions) correspond to inconsistent measurements. Any measurement made in a forbidden region is due to either ATE malfunction, or incorrect CANA modelling, or an undefined failure mode of the UUT.

It is desirable to transform each hypercube in the S-space to a single point in another space to simplify the fault diagnosis and isolation process. The designation number of each assertion may be interpreted as the number of unit distances of a region from an arbitrary origin along the measurement coordinate. This coordinate system is defined by the name of the corresponding diagnosis vector of the measurement. This transformation enables the N hypercubes in n -dimensional measurement space to be mapped to N_D points (D-point) in an n -dimensional space. The transformation space is called the diagnosis address space (D-space) of the diagnosis vectors.

It should be clear that any S-cube within an A-cube is also transformed to the same D-point in the address space. Each point in the D-space is identified by a unique address vector. When there are two or more failure modes which are mapped to the same D-point, the implication is that, with the given failure symptoms, these failures cannot be distinguished. Furthermore, if the D-point which is the image of the nominal mode is also the image of a failure mode, then this failure mode cannot be diagnosed. Since it is impossible to differentiate between the failure modes which map to the same point in the D-space (because they have the same address vectors) these failures constitute an equivalent failure class. Equivalent failure classes are defined by the end points of address vectors in the D-space. Figure 5.6 shows the address space transformation of Figure 5.5

Like the S-space, the D-space has allowed points which are used to diagnose failures. It also has unallowed points which indicate inconsistency in testing.

The symptom and address space interpretations of the failure symptoms are presented as the abstraction of the method used in this work to obtain diagnosis and fault isolation information from the tests conducted.

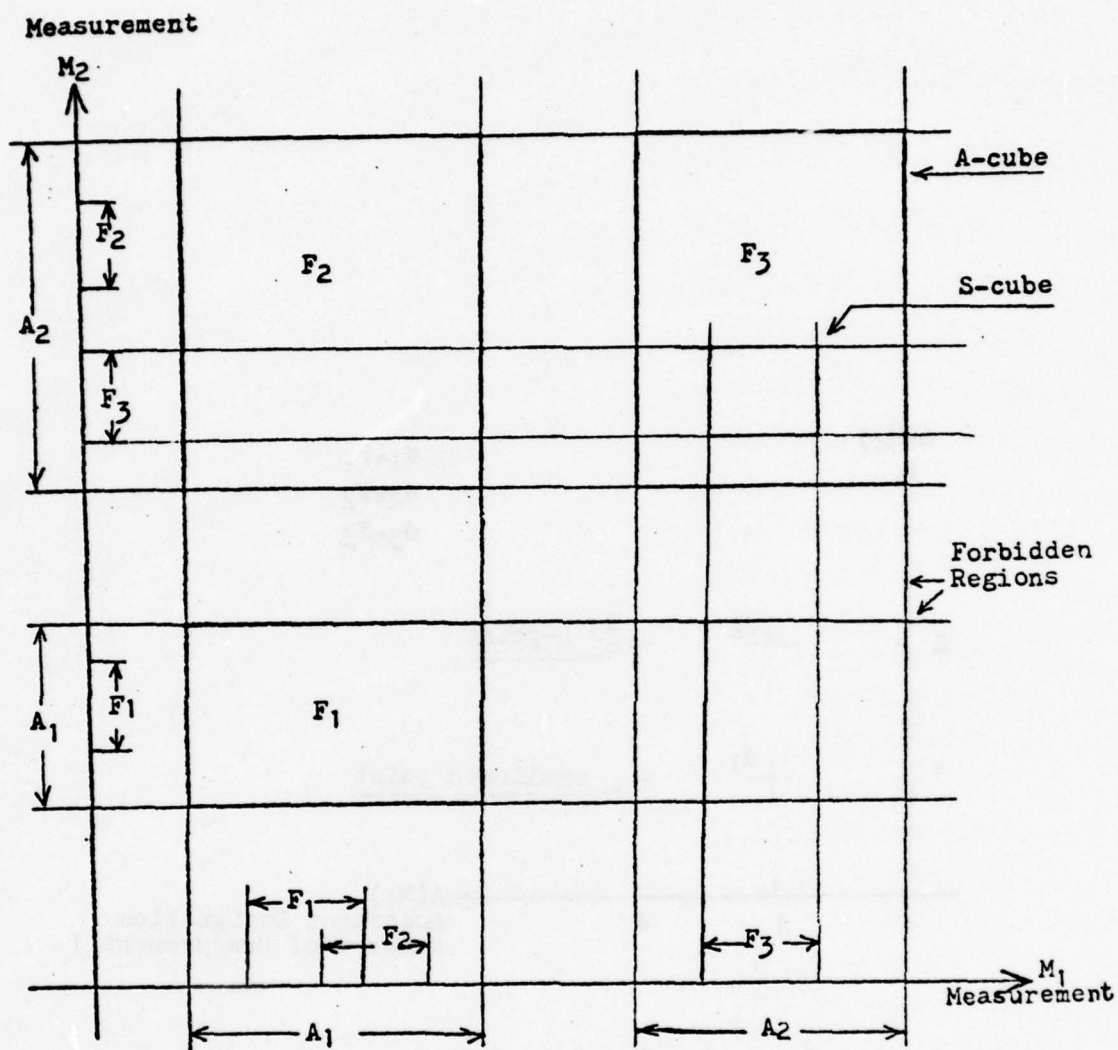


Figure 5.5 Symptom Space in 2-D

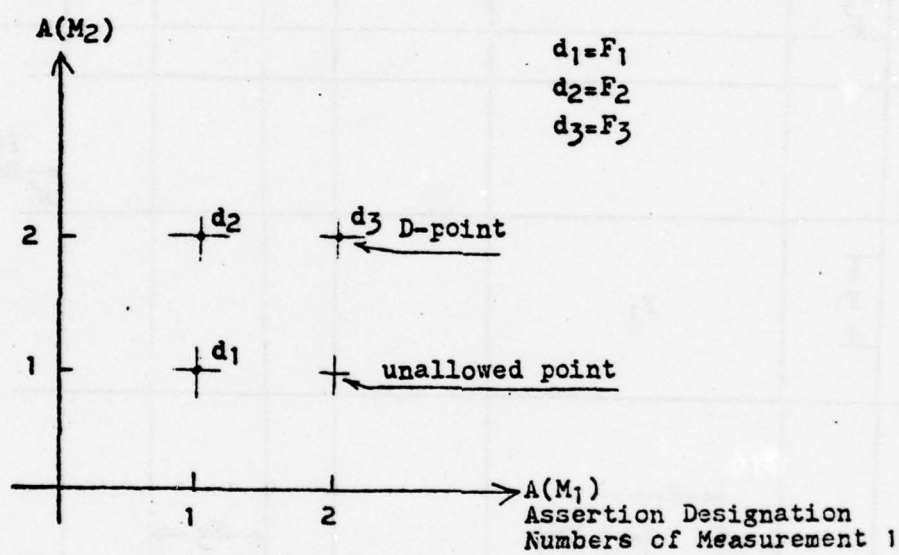


Figure 5.6 Address Space

5.7 A Method for Selecting Efficient Set of Tests

It is desirable to eliminate the tests and outcomes from a test specification, if they do not contribute to fault isolation. A figure-of-merit called information gain is used for this purpose. The use of information gain as a figure-of-merit (based on the analogy between optimum coding problem and diagnosis problem) was proposed by Brule et al. [107]. We shall explain the analogy from a similar approach.

In the mathematical model, an information source is associated with entropy H , which is thought of as the average amount of information contained in each symbol emitted by the source. Each test is an independent information source. The symbols emitted by the source are the outcomes of the corresponding tests. The capacity C of the channel is the amount of information required to isolate all failures. The information not getting through the channel is called the equivocation. In the diagnosis problem, this is the remaining ambiguity. The purpose of the test selection process is to achieve maximum diagnostic resolution using as few tests as possible. This is done by finding a number of sources which contribute the maximum entropy to the system. A source provides no information if it does not increase the entropy of the system.

Let N be the total number of failures. Assume that all failures are equally likely and that there can be only a single failure. Then, the probability of occurrence of any

failure is $P_j = 1/N_F$. The ambiguity before any evaluated test is:

$$\begin{aligned} \mathcal{H}_0 &= - \sum_{j=1}^{N_F} P_j \log_2 P_j \\ &= - N_F \left(\frac{1}{N_F} \log_2 \frac{1}{N_F} \right) \\ &= \log_2 N_F \end{aligned}$$

This is the required channel capacity and the amount of information needed for fault isolation. The application of a test reduces the ambiguity if the test contains non-zero information. That is, the information gained by the outcome of a test reduces ambiguity when the outcome cannot be predicted by other selected tests. The next best test to select is the one which reduces the ambiguity most; or in other words, the best test is the one which yields the highest average information gain:

$$\overline{\Delta \mathcal{H}}_k = \mathcal{H}_i - \sum_{j=1}^m \mathcal{H}_{i+1,j}$$

$\Delta \mathcal{H}_k$: entropy gain due to test T_k

\mathcal{H}_i : entropy due to tests $1, 2, \dots, i$

$\mathcal{H}_{i+1,j}$: entropy of equivalence class j determined by tests $1, 2, \dots, i, i+1$

\mathcal{H}_i is a constant; therefore $\overline{\Delta \mathcal{H}}_k$ is maximized at the maximum of $\sum_{j=1}^m \mathcal{H}_{i+1,j}$. The next test to be selected is T_k for which

$$\mathcal{H}_k = - \sum_{j=1}^m P_j \log_2 P_j$$

is maximum. Because we assumed that each failure is equally likely, the probability of observing equivalence class j is $p_j = \frac{n_j}{N_F}$, where n_j is the cardinality of the j^{th} equivalence class determined by the outcome of test T_k .

This process is repeatedly applied until the average information gain is zero or all available tests are used. The selected tests can still be applied in any sequence. It is clear that if a test is applied twice, the information gained from the second application does not increase the entropy of the system. This information measure is applicable to tests with binary or multiple valued outcomes. A result of information theory indicates that entropy is maximum when $p_1 = p_2 = \dots = p_m$. This property causes the selection of tests which tend to isolate failures into equivalence classes containing equal number of failures. This reduces the chances of making false diagnosis because each class will tend to have nearly equal number of failures.

CHAPTER 6

PROGRAM DESCRIPTION

This chapter presents a description of the programs found in the FITS system. The description is in sufficient detail which should make the reader comfortable enough to read the implementation in FORTRAN programming language.

The system design is described in five sections. They are: (1) system initialization, (2) failure simulation and symptom generation, (3) decision limits determination and ambiguity analysis, (4) optimization, and finally (5) NOPAL output generation.

In the system initialization section (Figure 6.1), the user input is scanned to create files which are used later by the system. They include the circuit description, failure dictionary, and the set of all applicable tests. A collection of statistics relating to the number of failure modes, number of tests and number of computer simulations, and a rough estimate of simulation time are also given.

In the failure simulation section (Figure 6.2), a subset of the applicable tests is simulated using a circuit analysis program. Then, the circuit analysis output is scanned to create the failure symptom table.

In the decision limit determination section (Figure 6.4), the failure symptoms are used to find easily

measurable regions of operation and malfunction. Then, the level of fault isolation possible by these decision limits is evaluated. If the results are satisfactory, the optimization process is initiated. Otherwise, simulation is resumed by trying the next set of candidate tests.

In the optimization section (Figure 6.5), a minimal number of test setups are selected, and the redundant assertions are eliminated without loss of fault isolation capability. Only the necessary assertions to select a diagnosis unambiguously are retained.

The last section (Figure 6.6) is related to the creation of the NOPAL table and the NOPAL specifications in string language.

6.1 System Initialization

The four programs in this section provide a convenient means to initialize the FITS system. They relate to user input processing, failure dictionary generation, printing of the failure dictionary, and generation of candidate tests libraries (Figure 6.1).

All system data files to be generated from the user's input are created as sequential files on the system storage devices. The purpose of this organization is to provide an external data base so that the user can tailor the system initialization for nonstandard usage. The more important files such as the failure dictionary are immediately printed. Other files may be printed on request. The user can make changes, additions, or deletions to these files either by rerunning the initialization program or by using the computer system's "editor" program.

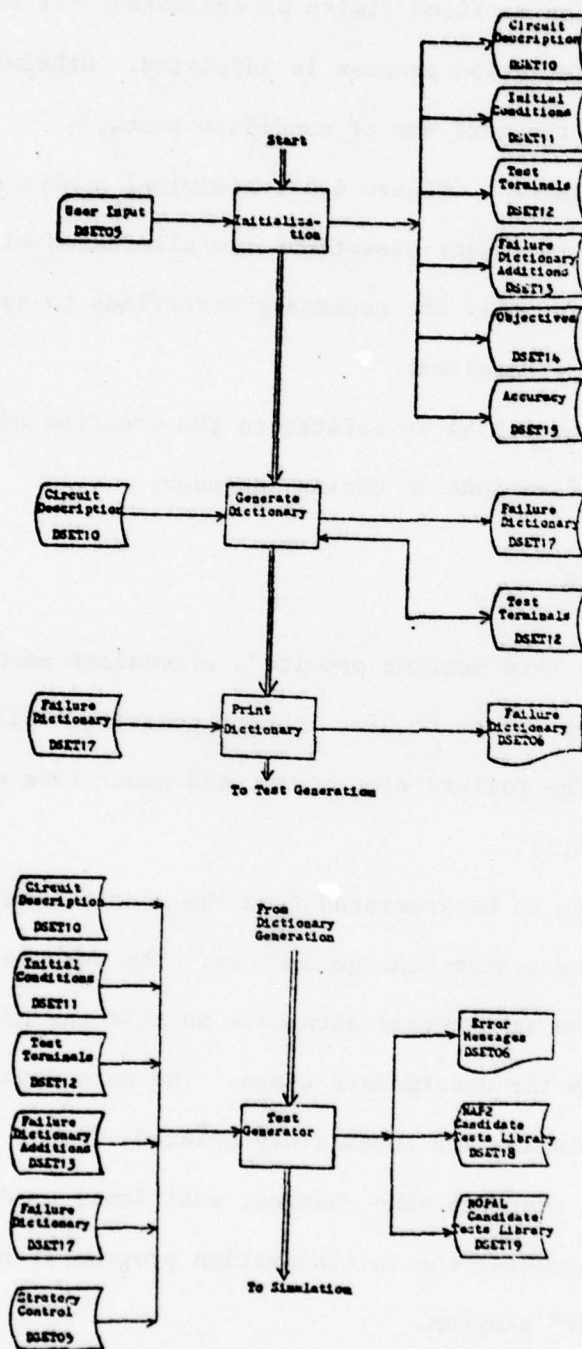


Figure 6.1 System Initialization

Program Name: INITIALIZE

Purpose: The program reads the user input from a single source and initializes each input section in a separate file.

Input: 1. (DSET05) User input

Output: 2. (DSET10) Circuit description
2. (DSET11) Initial conditions
3. (DSET12) Test terminals
4. (DSET13) Failure dictionary additions
5. (DSET14) Objectives
6. (DSET15) Accuracy

Data Structures: Free format user input (see Appendix A);
All input and output records are 80 characters long. The output records are exact duplicates of the input records.

Algorithm:

Step 1. Read (DSET05) and copy each input section to its own data set;

Step 2. For the input sections which are not specified generate the section identifier record and close all data sets;

Step 3. [End of program] Stop;

Program Name: GENERATE DICTIONARY

Purpose: The program reads the circuit description and generates a failure dictionary containing the catastrophic failures of the UUT. If no test terminals are specified, it also generates the default test terminals.

Input: 1. (DSET10) Circuit description
2. (DSET12) Test terminals

Output: 1. (DSET12) Test terminals
2. (DSET17) Failure dictionary

Data Structures:
(DSET17) Failure dictionary, see Table 6.1.

Algorithm:

- Step 1: Write (DSET17) identification section;
- Step 2: Read (DSET10) a record from the circuit description file; if end-of-file then go to Step 5;
- Step 3: If it is a component with failure, then generate its failure definition entries in DSET17;
- Step 4: Go to Step 2;
- Step 5: Close DSET17;
- Step 6: Read (DSET12) test terminals; if there are any test terminal statements, then go to Step 9;
- Step 7: Rewind DSET17 and write (DSET12) all circuit nodes in the dictionary as test terminals;
- Step 9: [End of program] Stop;

TABLE 6.1 DATA FIELDS IN FAILURE DICTIONARY

| NAME | COLUMN RANGE | DATA TYPE | DESCRIPTION |
|------------------|--------------|-------------------|---|
| ID. | 1-4 | integer (I4) | Failure mode sequence number |
| COMPONENT NAME | 5-16 | alphanum (3A4) | Name of the component as it appears in the circuit description |
| FAILURE FUNCTION | 17-20 | integer (I4) | Name of the failure function of the failure of the component |
| NODES | 21-32 | integer (3I4) | Identifies the incidence of the component on the circuit nodes |
| CHANGE | 33-36 | integer (I4) | Describes the change in the nominal circuit due to the failure |
| TYPE | 37-40 | integer (I4) | For simple failures identifies the new component type. Otherwise describes where the definition is to be found. |
| PARAMETER | 41-52 | alphanum (3A4) | Value of the new component or subcircuit library member name |
| VALUE | 53-64 | alphanum (3A4) | Value of the component in the nominal circuit |
| ALIAS | 65-76 | alphanum (3A4) | Alternate failure function name. This name is passed to the bottom part. |
| INDEX | 82-83 | integer (I2) | Failure rate index is not currently supported by FITS. If supplied, it is passed on to the bottom part. |

Program Name: PRINT DICTIONARY

Purpose: The program reads the failure dictionary and prints a user oriented tabular form of the internal representation.

Input: 1. (DSET17) Failure dictionary

Output: 1. (DSET06) Printer output
2. (DSET50) Work file

Algorithm:

Step 1: Print failure dictionary identification;

Step 2: Read (DSET17) a failure definition; if end-of-file, then go to Step 6;

Step 3: Convert internal representation of symbols to external representation;

Step 4: Print the external representation of the failure definition;

Step 5: Go to Step 2;

Step 6: [End of program] Stop;

Program Name: TEST GENERATOR

Purpose: The program generates all candidate tests as partially specified NOPAL test modules and as complete NAP2 input.

Input:

1. (DSET05) Strategy control
2. (DSET10) Circuit description
3. (DSET11) Initial conditions
4. (DSET12) Test terminals
5. (DSET13) Failure dictionary additions
6. (DSET17) Failure dictionary

Output:

1. (DSET06) Printer output (error messages)
2. (DSET18) NAP2 candidate tests library
3. (DSET19) NOPAL candidate tests library
4. (DSET50) Work file

Data Structures:

- (DSET05) Strategy control, see Table 6.2
- (DSET18) 80-byte records written in NAP2 input language.
- (DSET19) 80-byte records written in NOPAL language.

Algorithm:

Step 1. Write (DSET18) NAP2 candidate tests library identification;

Step 2. Write (DSET19) NOPAL candidate tests library identification;

Step 3. Read (DSET05) test generator control options;

Step 4. Read (DSET12) test terminals;

Step 5. Read (DSET17) failure dictionary;

Step 6. If cold circuit strategy tests are desired, then call subprogram CCTEST;

Step 7. If direct current and user defined strategy tests are desired, then call subprogram DCTEST;

Step 8. Close files DSET18 and DSET19;

Step 9. Print types of strategies used, number of tests generated, and expected simulation time;

Step 10. [Test generation completed] Stop;

TABLE 6.2 TEST STRATEGY CONTROL OPTIONS

| NAME | COLUMN RANGE | DEFAULT | DESCRIPTION |
|--------|--------------|---------|--|
| STRTRY | 1 | 0 | 0 -- CC, DC and UD strategy tests 1 -- CC strategy tests only 2 -- DC and UD strategy tests |
| OUTPUT | 2 | 0 | 0 -- both NAP2 and NOPAL 1 -- NAP2 only 2 -- NOPAL only |
| ONLY | 3-6 | <N > | <N > number of failures to be simulated. If <N >=1, then only the nominal circuit description is in simulations. |
| SEQTM | 7-10 | 0 | last test module sequence number |
| SEQSTM | 11-14 | 0 | last stimulus sequence number |
| SEQMSR | 15-18 | 0 | last measurement sequence number |

Subprogram Name: CCTEST

Purpose: This subprogram generates only the cold circuit strategy tests in NAP2 and NOPAL languages. Cold circuit tests apply a small current source shunted by a small conductance across all pairwise combinations of test terminals. The worst-case voltage measurement across the test terminals where the current source is applied is used to estimate the dc-impedance of the UUT between these terminals. Because the magnitude of the applied current is extremely small it is not applied at reversed polarity.

Algorithm:

- Step 1. For all combinations of test terminals taken two at a time do Steps 2 through 3;
- Step 2. Call subprogram CCNOPL to write the test specification in NOPAL;
- Step 3. Call subprogram CCNAP2 for each failure mode in the failure dictionary to generate the NAP2 statements to simulate the application of an ohmmeter;
- Step 4. Return to main program;

Subprogram Name: DCTEST

Purpose: The program generates the direct current and user defined tests in NAP2 and NOPAL languages. In the DC test strategy instructions are written to connect the power supplies of the UUT. Then instructions are generated to calculate the currents through the voltage sources, the voltages across the current sources and the voltage level at the remaining test terminals. If there are any user defined test strategies they are also processed. If there are any conditions which the user wants to be calculated to check for overstressing, they too may be requested for output. However, these results are ignored by the FITS programs.

Algorithm:

- Step 1. Input (DSET11) an initial condition; if end-of-file, then return to main program;
- Step 2. If the initial condition involves a user defined strategy, then go to Step 8;
- Step 3. [DC strategy] For all power supplies of the circuit, do Step 4;
- Step 4. If the power supply is a voltage source, then measure the current through the voltage supply;
 - 4.1 Call subprogram DCNOPL to write the test specification in NOPAL;
 - 4.2 Call subprogram DCTYP1 to write the test specifications for each failure definition in the dictionary;
- Step 5. If the power supply is a current source, then measure the voltage across the connecting terminals; call subprograms DCNOPL and DCTYP2;
- Step 6. For all test terminals which are not incident on power supplies, measure the voltage across the test terminal and ground; call subprograms DCNOPL and DCTYP3;
- Step 7. Go to Step 1;
- Step 8. Separate the NAP2 and NOPAL statements and write them into their files; repeat the test for each failure in the dictionary;
- Step 9. [Start with a new initial condition] Go to Step 1;

6.2 Failure Simulation and Symptom Generation

The input prepared by the candidate tests program is analyzed by the NAP2 circuit analysis program. The design of this stage treats the CANA system as a black box and uses it without any internal changes of the source code. In spite of the fact that time and memory may be saved by a special purpose CANA package for FITS, it was chosen to incorporate NAP2 as it is available to any other user. Hence, improvements made to the NAP2 program can be used without necessitating changes in the FITS system. As a matter of fact, NAP2 may be replaced altogether by another CANA package after properly modifying the syntactical portions of the candidate tests generator program and the failure symptom generator program. No changes in the remaining programs of the FITS system would be necessary.

The second program searches through the simulation output and creates the failure symptom program (See Figure 6.2).

Program Name: NAP2

Purpose: The program simulates the tests for the nominal and the failed UUT to determine the failure symptoms at the indicated test terminals.

Input: 1. (DSET05) NAP2 candidate tests library

Output: 2. (DSET06) NAP2 simulation output

Data Structures: (DSET06) is the printer output of NAP2. Each output record is 132 bytes long. For the format of output records see the examples in Appendix A.

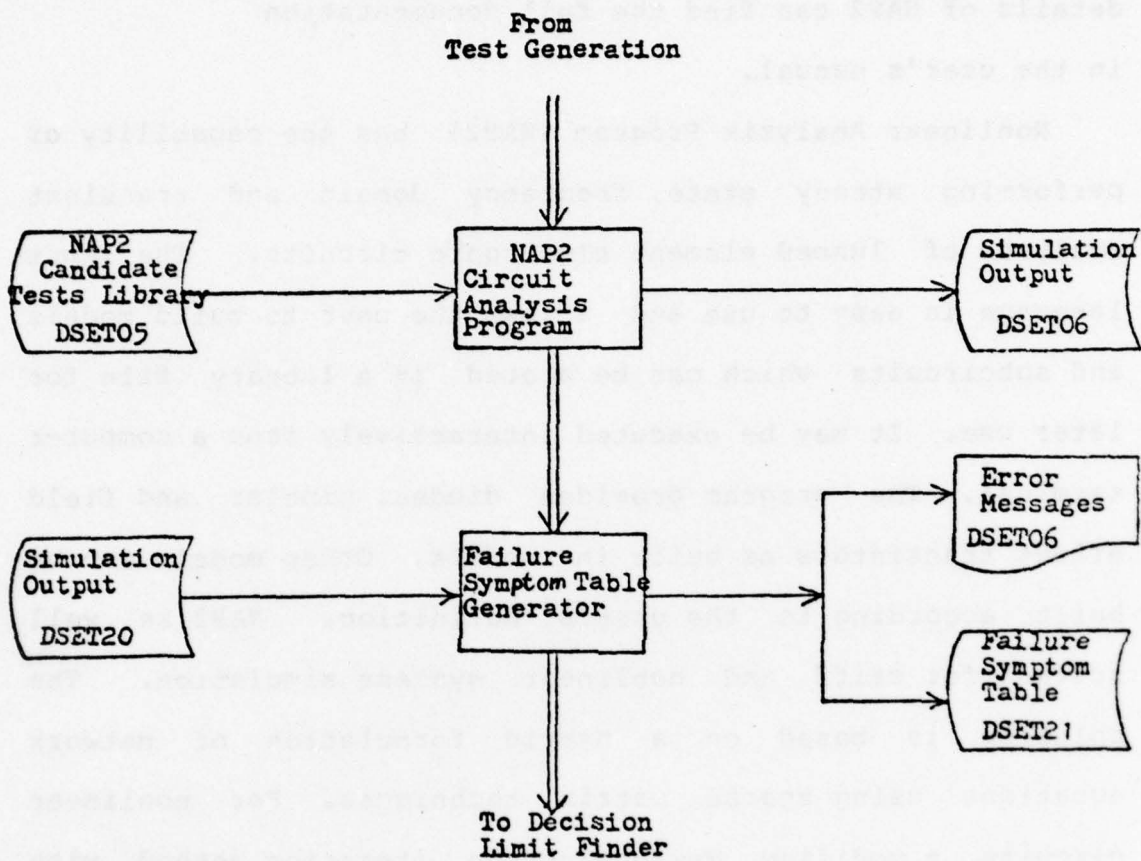


Figure 6.2 Failure Simulation and Symptom Generation

Figure 6.3 shows the program structure of the NAP2 circuit analysis program. Here only a general description of the program is given. The readers interested in the details of NAP2 can find the full documentation in the user's manual.

Nonlinear Analysis Program (NAP2) has the capability of performing steady state, frequency domain and transient analyses of lumped element electronic circuits. The input language is easy to use and allows the user to build models and subcircuits which can be stored in a library file for later use. It may be executed interactively from a computer terminal. The program provides diodes, bipolar and field effect transistors as built in models. Other models can be built according to the user's definition. NAP2 is well suited for stiff and nonlinear systems simulation. The solution is based on a hybrid formulation of network equations using sparse matrix techniques. For nonlinear circuits a modified Newton-Raphson iteration method with damping is used. In transient analysis an implicit, variable-order, variable-step integration scheme is used. A constant step size and/or fixed order integration may be selected. Network sensitivities are computed from adjoint network in the steady state and frequency domain analysis, while the time dependent sensitivities are calculated directly from the difference equations produced by the integration formulii. All circuit parameters may be functionally dependent on other parameters or on the circuit

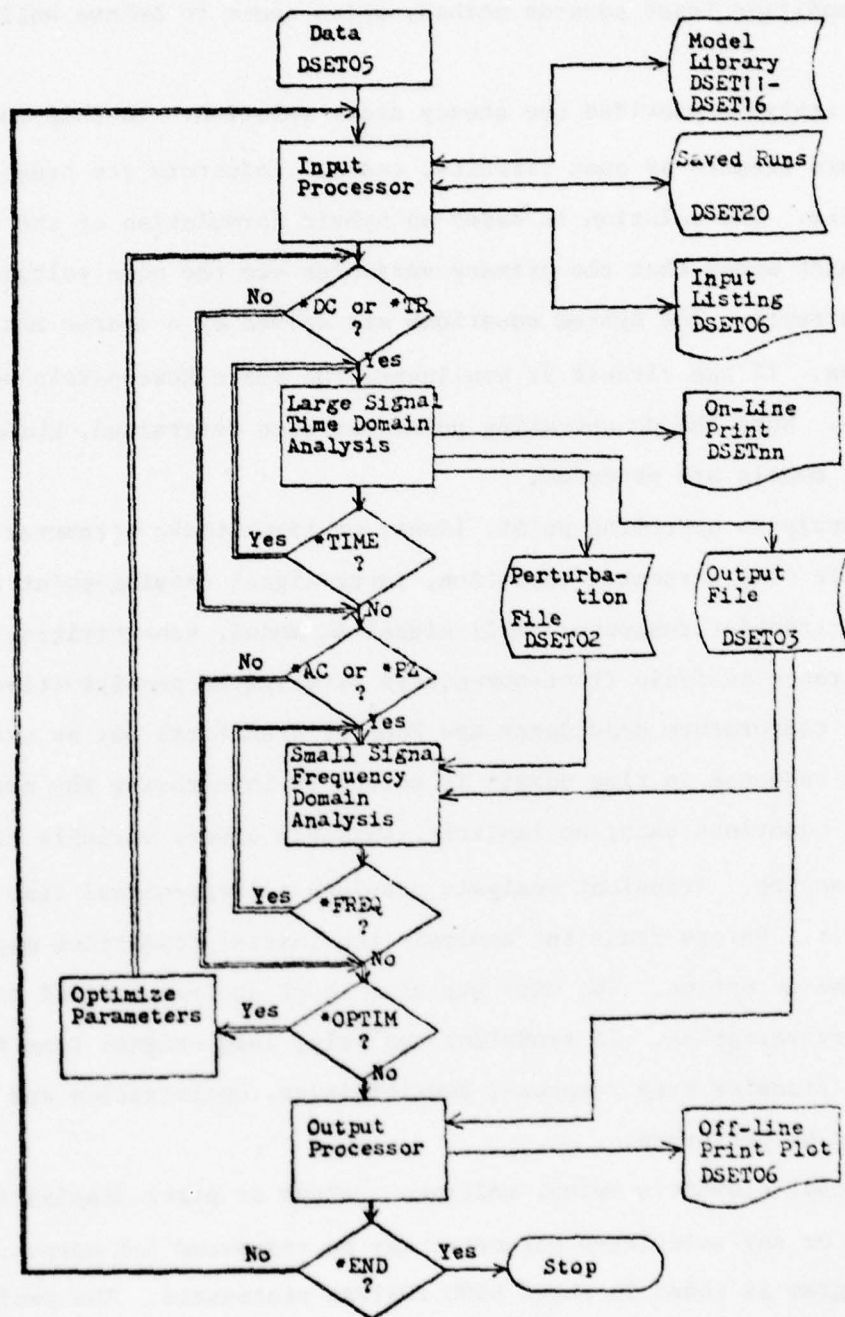


Figure 6.3 NAP2 Program Structure

responses. General nonlinearities are modelled in this fashion. Automatic design is made possible by the use of an optimization loop. This technique is based on a modified least squares method, which seems to behave well for some problems.

The dc analysis provides the steady state solution. In this mode all capacitors are treated as open circuits, and all inductors are treated as short circuits. The solution is based on hybrid formulation of the network equations which means that the primary variables are the node voltages and the impedance currents. The system equations are solved by a sparse matrix LU decomposition. If the circuit is nonlinear, the basic Newton-Ralphson iteration is used. When the dc operating point has been determined, linearized small-signal models are produced.

In dc analysis operating point, linear or logarithmic parameter variation, variable step parameter variation, large-signal driving-point response, small-signal transfer response, small-signal AC model, sensitivities, optimization, tolerance analysis (root-sum-square of weighted sensitivities), worst-case, noise, temperature dependency and Fourier transforms may be performed.

Network response in time domain is solved by integrating the network differential equations using an implicit, variable order, variable step size integration scheme. Transient analysis provides a large-signal time response of the circuit. Before transient analysis the initial conditions may be calculated by the dc option. The user may also input desired initial conditions to the primary variables. In transient analysis, large-signal time response, small-signal transfer step response, sensitivities, optimization and Fourier transform may be performed.

Any circuit element's value, voltage, current or power dissipation, any node voltage or any calculated parameter may be requested for output.

The program is coded in about 6500 FORTRAN statements. The small version (100 kilobytes) can analyze circuits with 50 nodes and 200 branches, and the bigger version can analyze circuits containing up to 500 nodes and 1500 branches.

NAP2 was developed by the Northern Europe University Computing Center (NEUCC), Technical University of Denmark in Lyngby.

Program Name: FAILURE SYMPTOM TABLE GENERATOR

Purpose: The program searches the NAP2 simulation output and extracts only the data which is used by the FITS system. The purpose of this process is to communicate the circuit analysis results to the remainder of the FITS system. It searches the bulky simulation results and extracts all significant data of failure analysis. This information is put into the failure symptom table. If during the search process "NO CONVERGENCE" messages are detected they are printed separately with proper identification. The user is then expected to remove these conditions by providing better initial conditions to restart the Newton-Raphson iteration.

Input: 1. (DSET20) NAP2 simulation output

Output: 1. (DSET06) Printer output (convergence problems)
2. (DSET21) Failure symptom table

Data Structures: (DSET21) See Table 6.3

Algorithm:

- Step 1. Read (DSET20) the next run identification; if end-of-file then stop;
- Step 2. Find the nominal, minimum and maximum worst-case measurements and sensitivity values;
- Step 3. Write the record on DSET21;
- Step 4. If 'NO CONVERGENCE' message is found, then display the run identification on the printer;
- Step 5. [Process the next failure symptom] Go to Step 1;

TABLE 6.3 DATA FIELDS IN THE FAILURE SYMPTOM TABLE

| NAME | COLUMN RANGE | DESCRIPTION |
|-------|--------------|--|
| STIM | 2-5 | stimulus sequence number |
| MEAS | 6-9 | measurement sequence number |
| FAULT | 10-13 | failure sequence number |
| NAME | 17-28 | target variable name (identifies the measurements) |
| NOM | 29-42 | nominal measurement |
| MIN | 45-58 | minimum worst-case measurement |
| MAX | 61-73 | maximum worst-case measurement |
| SNOM | 76-88 | root sum square (RSS) sensitivity of nominal measurement |
| SMIN | 89-101 | RSS sensitivity of minimum worst-case measurement |
| SMAX | 102-114 | RSS sensitivity of maximum worst-case measurement |

6.3 Decision Limits Determination and Ambiguity Analysis

The first program in this section reads the failure symptom table and generates the assertions from each test while taking the measurement inaccuracy into consideration.

In order to evaluate the level of fault isolation achieved with the information accumulated up to this stage, a simple ambiguity analysis is performed. The results of ambiguity analysis are compared to the test objectives set forth by the user (see Figure 6.4). If the fault isolation objectives are not satisfactory some more of the tests are simulated and evaluated similarly. If the objectives are satisfied, then the optimization process is started.

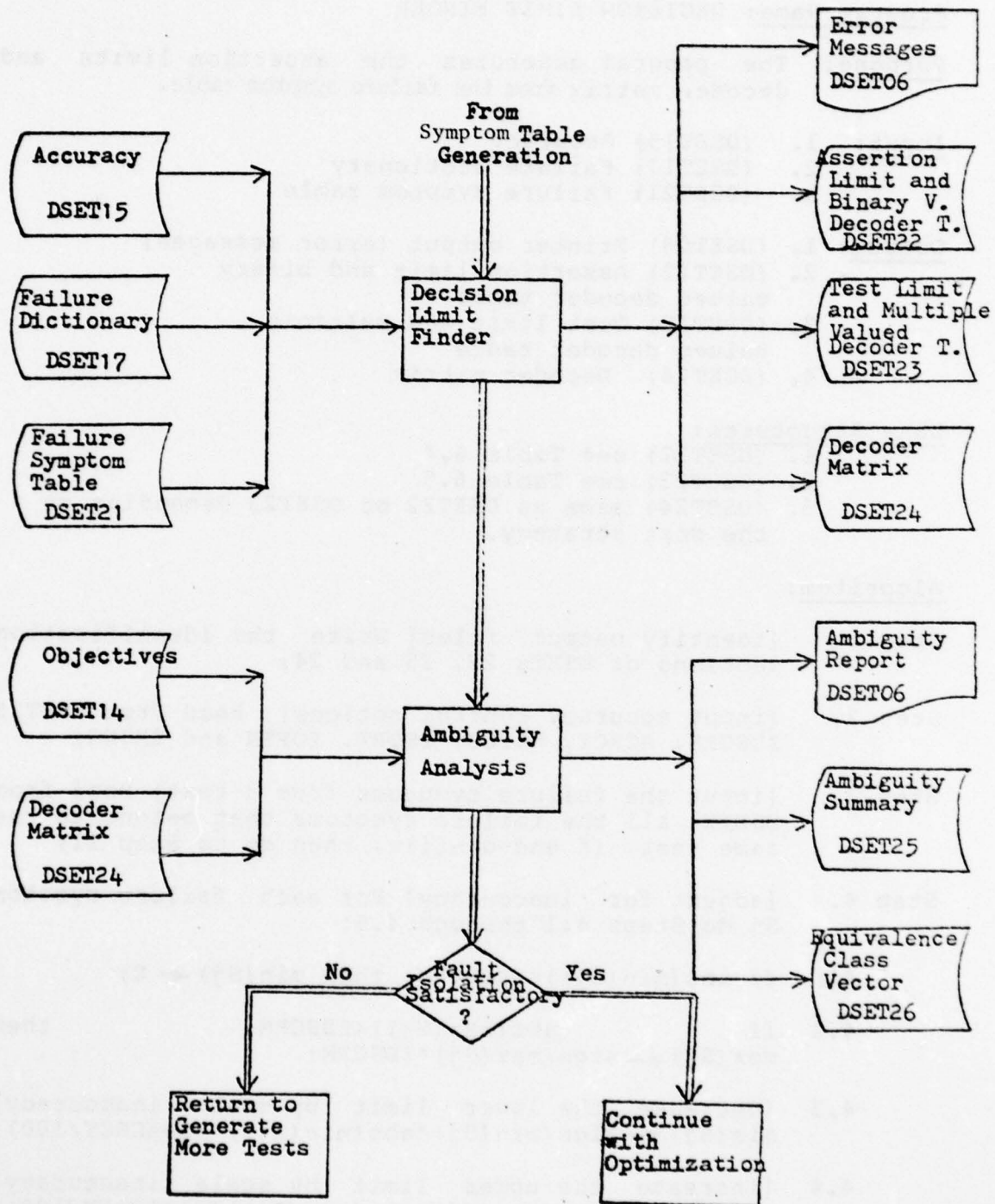


Figure 6.4 Decision Limit Determination and Ambiguity Analysis

Program Name: DECISION LIMIT FINDER

Purpose: The program generates the assertion limits and decoder matrix from the failure symptom table.

Input:

1. (DSET15) Accuracy
2. (DSET17) Failure Dictionary
3. (DSET21) Failure symptom table

Output:

1. (DSET06) Printer output (error messages)
2. (DSET22) Assertion limit and binary valued decoder table
3. (DSET23) Test limit and multiple valued decoder table
4. (DSET24) Decoder matrix

Data Structures:

1. (DSET22) see Table 6.4
2. (DSET23) see Table 6.5
3. (DSET24) same as DSET22 or DSET23 depending on the sort strategy.

Algorithm:

Step 1. [identify output files] Write the identification sections of DSETs 22, 23 and 24;

Step 2. [input accuracy control options] Read from DSET15 ZDSCRM, ACRCY, NSIGD, ISORT, IOPTM and IMSNG;

Step 3. [input the failure symptoms from a test] Read from DSET21 all the failure symptoms that belong to the same test; if end-of-file, then go to Step 11;

Step 4. [adjust for inaccuracy] For each failure symptom S_j do Steps 4.1 through 4.5:

4.1 If $\text{abs}(\min(S_j)) < \text{ZDSCRM}$, then $\min(S_j) \leftarrow 0$;

4.2 If $\text{abs}(\max(S_j)) < \text{ZDSCRM}$, then $\max(S_j) \leftarrow \text{sign}(\max(S_j)) * \text{ZDSCRM}$;

4.3 [decrease the lower limit by scale inaccuracy] $\min(S_j) \leftarrow \text{sign}(\min(S_j)) * \text{abs}(\min(S_j)) * (1 - \text{ACRCY}/100)$;

4.4 [increase the upper limit by scale inaccuracy] $\max(S_j) \leftarrow \text{sign}(\max(S_j)) * \text{abs}(\max(S_j)) * (1 + \text{ACRCY}/100)$;

4.5 [retain significant digits] Round up upper limit to NSIGD, and round down lower limit to NSIGD;

Step 5. [process missing failure symptoms]

5.1 Read in the CHANGE options from the failure

dictionary (DSET17);

- 5.2 Delete the failure symptoms which have been declared "delete" in the change options;
 - 5.3 Duplicate the symptoms of failures which are declared to be same as another failure;
 - 5.4 Copy the symptoms of the failure sequence number IMSNG into the missing failure symptoms;
- Step 6. [create assertions] Find the nonoverlapping ranges of measurements and identify the failures which fall into these ranges;
- Step 7. [create binary valued decoder matrix] Create a row for each assertion in the BVDM where the columns are labelled by failure sequence numbers: put 1 under the column of a failure which has its symptom in the range of this assertion, otherwise put 0;
- Step 8. [create multiple valued decoder vector] For the current test create the MVDV by placing the designation number of the assertion in the column of the failure which has its symptom in that assertion range;
- Step 9. [calculate the average sensitivities and standard deviation] For each assertion and test calculate the root sum square average and the standard deviation of the sensitivities of the failure symptoms; output the result on DSET22 and DSET23;
- Step 10. [process the next test] Go to Step 1;
- Step 11. [sort assertions and/or tests] If the sort of assertions and tests are requested sort them by calling the bubble sort subprogram BUBSRT according to increasing sensitivity order;
- Step 12. [print] Print DSET22 and DSET24;
- Step 13. [end of program] Stop;

AD-A056 660

MOORE SCHOOL OF ELECTRICAL ENGINEERING PHILADELPHIA P--ETC F/G 9/3
AUTOMATED TEST DESIGN. (U)

JUN 78 C TINAZTEPE

DAAA25-75-C-0650

UNCLASSIFIED

77-03

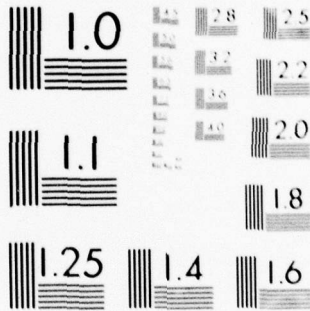
ECOM-75-0650-F-2

NL

3 of 4

AD
A056 660





MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

TABLE 6.4 DATA FIELDS IN ASSERTION LIMIT
AND BINARY VALUED DECODER TABLE

| NAME | COLUMN RANGE | DESCRIPTION |
|----------------------|---------------------|--|
| SEQ | 1-4 | row count of the assertion |
| STIM | 5-8 | stimulus sequence number |
| MEAS | 9-12 | measurement sequence number |
| TARGET | 13-24 | target variable name |
| DESIG | 25-28 | assertion designation number |
| LOWER LIMIT | 29-43 | assertion lower limit |
| UPPER LIMIT | 59-73 | assertion upper limit |
| SENSITIVITY (RSS) | 74-88 | root sum square of the sensitivities of the failure symptoms found in this assertion range |
| SENSITIVITY (STD) | 89-103 | standard deviation of the sensitivities of the failure symptoms found in this assertion range |
| BVDV | 104-303 (100*12) | binary valued diagnosis vector showing which failures have their symptoms in this assertion (1 indicates in range, 0 indicates out of range) |

TABLE 6.5 DATA FIELDS IN THE TEST LIMIT AND
MULTIPLE VALUED DECODER TABLE

| NAME | COLUMN RANGE | DESCRIPTION |
|-------------------|---------------------|---|
| SEQ | 1-4 | pointer to the first row of the assertions that belong to this test in the Assertion Limit and Binary Valued Decoder Table |
| STIM | 5-8 | stimulus sequence number |
| MEAS | 9-12 | measurement sequence number |
| TARGET | 13-24 | target variable name |
| DESIG | 25-38 | total number of assertions of this test |
| LOWER LIMIT | 29-43 | absolute minimum measurement possible due to any failure |
| UPPER LIMIT | 59-73 | absolute maximum measurement possible due to any failure |
| SENSITIVITY (RSS) | 74-88 | root sum square of the sensitivities of all of the symptoms of failures |
| SENSITIVITY (STD) | 89-103 | standard deviation of the sensitivities of all of the symptoms of failures |
| MVDV | 104-303 (100*12) | multiple valued diagnosis vector, each entry contains the designation number of the assertion which contains the symptom of failure |

Program Name: AMBIGUITY ANALYSIS

Purpose: The program finds the faults which can not be distinguished from one another with the available tests. It produces an ambiguity report showing the level of fault isolation possible with the current decoder matrix.

Input: 1. (DSET14) Objectives
2. (DSET24) Decoder matrix

Output: 1. (DSET06) Printer output (ambiguity report)
2. (DSET25) Ambiguity summary
3. (DSET26) Equivalence class vector

Data Structures:

(DSET25) Tabular listing of ambiguity summary, format free.

(DSET26) 100 integers each occupying 2 columns positionally indicating which equivalence class a failure belongs to.

Algorithm:

- Step 1. [identify output files] Write identification section of DSET25 and DSET26;
- Step 2. [input fault isolation objectives] Read objectives from DSET14;
- Step 3. [input MVDM] Read multiple valued decoder matrix from DSET24;
- Step 4. [identify distinct address vectors] Call subprogram LEAVES to determine the distinct address vectors and the corresponding failures;
- Step 5. [print results] Print the ambiguity report and summary on DSET06;
- Step 6. [save equivalence class names vector] Write the equivalence class vector on DSET26;
- Step 7. [end of ambiguity analysis] Stop;

6.4 Optimization to Minimize Test Lengths

There are three programs of this section (see Figure 6.4). The first program "Optimize Setup or Assertion" serves two purposes. If the input is a multiple valued decoder matrix, the number of different test setups is minimized. However, if the input is a binary valued decoder matrix, then the number of assertions is minimized. In case the multiple valued decoder matrix is optimized, then the program called "Transform Multiple to Binary" must also be executed if the number of assertions are to be optimized. The third program called "Optimize Assertions per Diagnosis" must be executed before NOPAL output processing. The optimization process in this program may be omitted. The input may be multiple or binary valued decoder matrix. The output is the diagnosis matrix which is used in NOPAL generation.

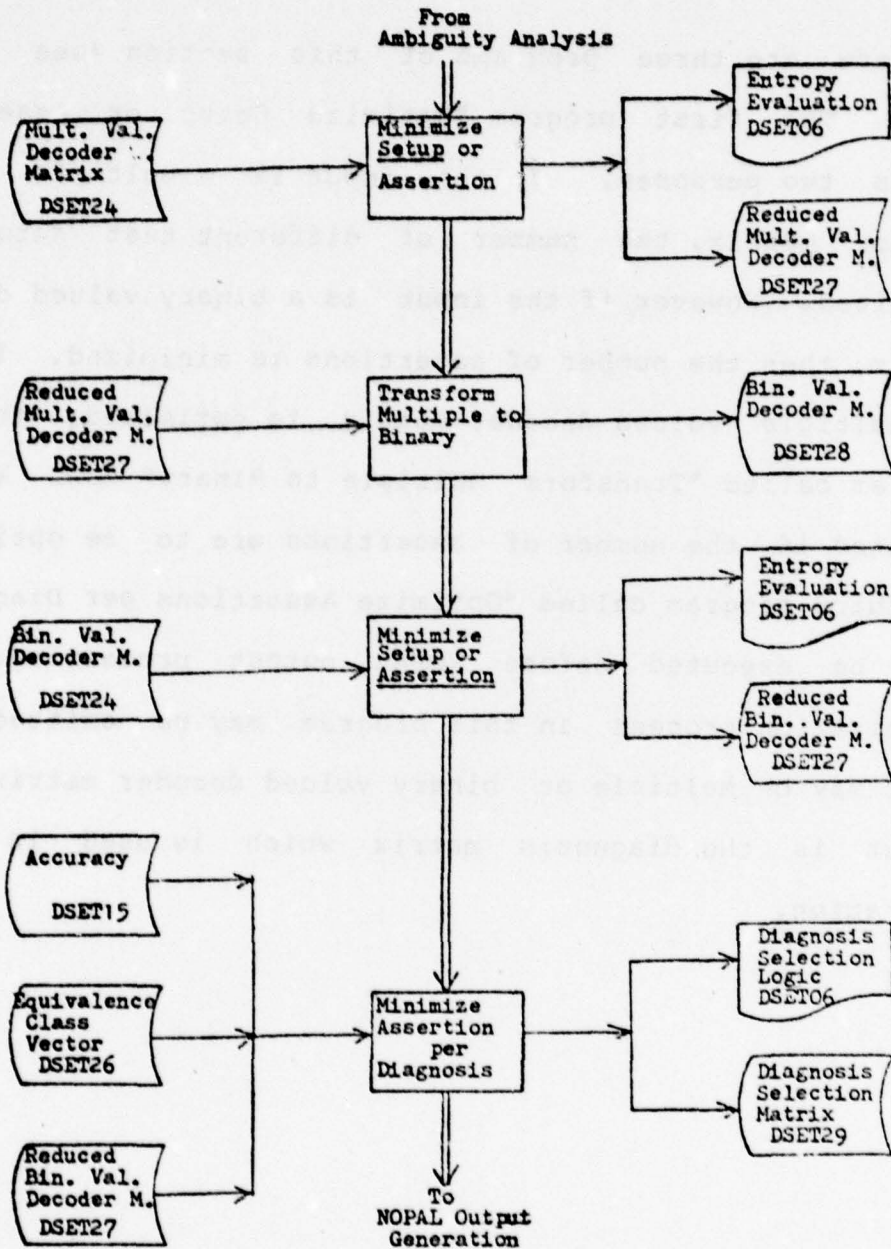


Figure 6.5 Optimization to Minimize Test Length

Program Name: OPTIMIZE SETUP OR ASSERTION

Purpose: The program finds a subset of the test setups or assertions which can yield the same level of fault isolation as the complete set can.

Input: 1. (DSET24) Decoder matrix

Output: 1. (DSET06) Printer output (entropy evaluation)
2. (DSET27) Reduced decoder matrix

Data Structure:

DSET27 is organized same as DSET24

Algorithm:

- Step 1. [identify output file] Write the identification section of DSET27;
- Step 2. [input test identification and decoder matrix] Read ID and DM from DSET24;
- Step 3. [find the test with maximum entropy] Call subprogram GENCDV to calculate the entropy for the tests not already included in the tree. Include the test with the highest entropy in the next level of the fault isolation tree;
- Step 4. [termination check] If all of the tests are included in the tree or the entropy has not increased by the inclusion of the next then go to Step 5 else go to Step 3;
- Step 5. [print summary] Print the entropy of each individual test and the cumulative entropy at each level of the fault isolation tree;
- Step 6. [end of program] Stop;

Program Name: TRANSFORM MULTIPLE TO BINARY

Purpose: The program converts the multiple valued decoder matrix to a binary valued decoder matrix. It avoids a time-costly search through the assertion limit and binary valued decoder table.

Input: 1. (DSET27) Multiple valued decoder matrix

Output: 1. (DSET28) Binary valued decoder matrix

Algorithm:

- Step 1. [input a multiple valued decoder vector] Read a MVDV from DSET27; if end-of-file, then stop;
- Step 2. [generate binary valued vectors] Do Steps 3 through 4 for i from 1 until ASSERT (where ASSERT is the number of assertion of the test);
- Step 3. For j from 1 until N_F (where N_F is the number of failures) ;if $MVDV(j)=i$, then $BVDV(i) \leftarrow 1$ else $BVDV(i) \leftarrow 0$;
- Step 4. [output BVDV] Write BVDV to DSET28;
- Step 5. [continue with the next MVDV] Go to Step 1;

Program Name: MINIMIZE ASSERTIONS PER DIAGNOSIS

Purpose: The program produces minimum length test specifications for each equivalent group of failures.

Input: 1. (DSET15) Accuracy
2. (DSET26) Equivalence class vector
3. (DSET27) Decoder matrix

Output: 1. (DSET06) Printer output (summary)
2. (DSET29) Diagnosis matrix

Data Structures: (DSET29) see Table 6.6

Algorithm:

- Step 1. Write the identification section of DSET29;
- Step 2. Read IOPTM control from DSET15;
- Step 3. Input equivalence class vector from DSET26;
- Step 4. Input decoder matrix from DSET27;
- Step 5. Compress the decoder matrix using the equivalence class vector such that all columns are distinct;
- Step 6. If IOPTM=0, then go to Step 8;
- Step 7. Find the minimum length join of tests or assertions to distinguish one diagnosis from another;
- Step 8. Calculate and print the test length statistics;
- Step 9. If the input decoder matrix was multiple valued, transform it to binary valued;
- Step 10. [write the diagnosis selection logic matrices]
Write DM (selection by disjunctions) and CM (selection by conjunctions) to DSET29;
- Step 11. [end of program] Stop;

TABLE 6.6 DATA FIELDS IN THE DIAGNOSIS MATRIX

| NAME | COLUMN RANGE | DESCRIPTION |
|--------|---------------------|--|
| SEQ | 1-4 | row number of the diagnosis in the assertion limit and binary valued decoder matrix |
| STIM | 5-8 | stimulus sequence number |
| MEAS | 9-12 | measurement sequence number |
| ASSERT | 13-16 | assertion designation number |
| DM | 17-216 (100*12) | diagnosis selection by disjunctions (0 means blank, 1 means selection by disjunctions. The columns are labelled by the distinct equivalence classes |
| CM | 217-416 (100*12) | diagnosis selection by conjunctions (0 means blank, 1 means selection by conjunction "&", and 2 means selection by negated conjunction (&-). The columns are labelled by the distinct equivalence classes. |

6.5 Nopal Output Generation

There are two programs in this section (see Figure 6.6). The first program generates the tabular NOPAL specification. The second program generates the complete NOPAL specifications.

The tabular NOPAL specification is intended to be a convenience for the reader of the NOPAL specification. In this table the emphasis is on the diagnosis selection logic section of the test modules. It helps the user to see the "sparseness" of the diagnosis selection.

The output of the second program is in a form which can be directly given to the bottom part of the NOPAL processor.

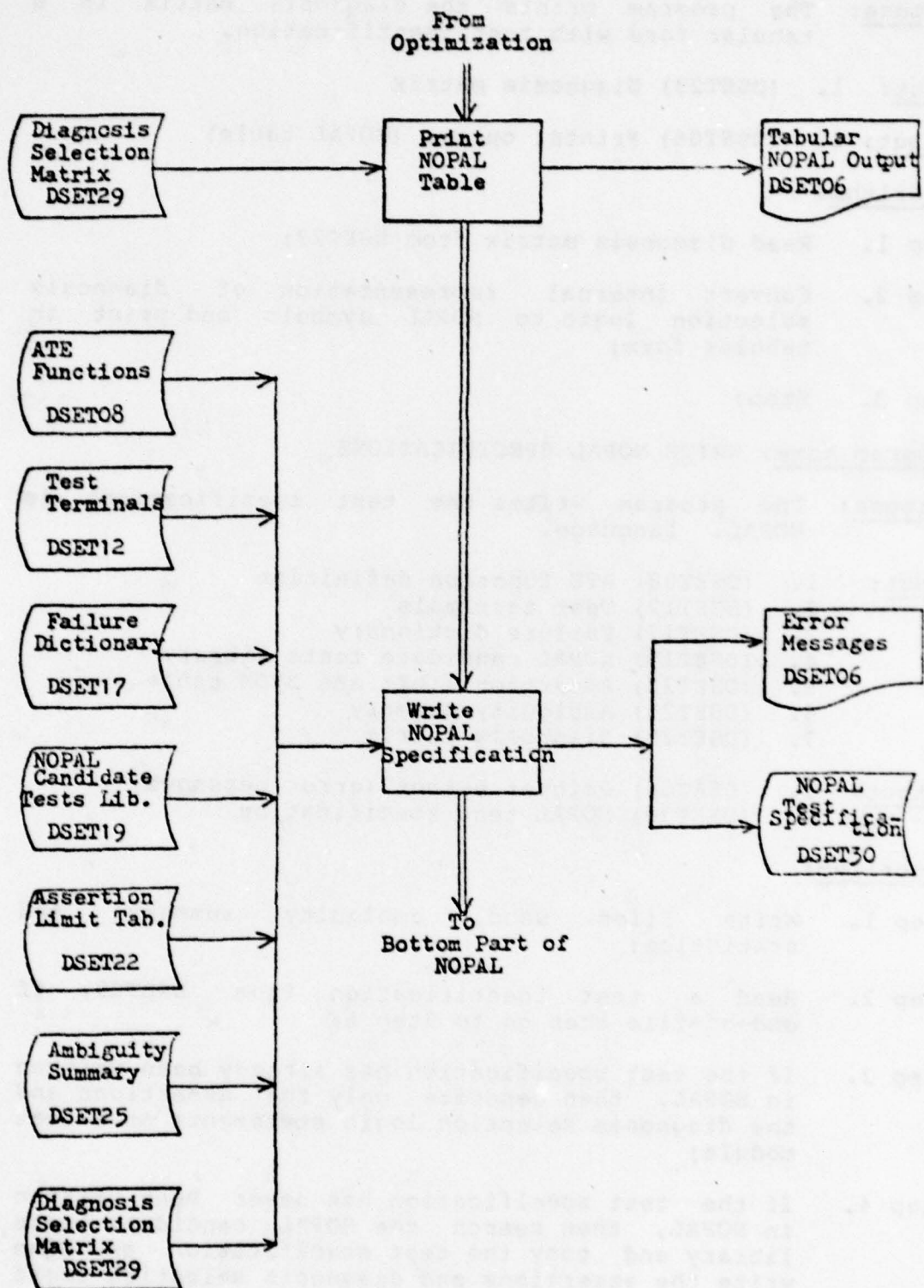


Figure 6.6 NOPAL Output Generation

Program Name: PRINT NOPAL TABLE

Purpose: The program prints the diagnosis matrix in a tabular form with test identification.

Input: 1. (DSET29) Diagnosis matrix

Output: 1. (DSET06) Printer output (NOPAL table)

Algorithm:

Step 1. Read diagnosis matrix from DSET29;

Step 2. Convert internal representation of diagnosis selection logic to NOPAL symbols and print in tabular form;

Step 3. Stop;

Program Name: WRITE NOPAL SPECIFICATIONS

Purpose: The program writes the test specifications in NOPAL. language.

Input:

1. (DSET08) ATE function definition
2. (DSET12) Test terminals
3. (DSET17) Failure dictionary
4. (DSET19) NOPAL candidate tests library
5. (DSET22) Assertion limit and BVDM table
6. (DSET25) Ambiguity summary
7. (DSET29) Diagnosis matrix

Output:

1. (DSET06) Printer output (error messages)
2. (DSET30) NOPAL test specification

Algorithm:

Step 1. Write files used, ambiguity summary, and statistics;

Step 2. Read a test identification from DSET29; if end-of-file then go to Step 6;

Step 3. If the test specification has already been written in NOPAL, then generate only the assertions and the diagnosis selection logic statements as a test module;

Step 4. If the test specification has never been written in NOPAL, then search the NOPAL candidate tests library and copy the test specification and also write the assertions and diagnosis selection logic statements in NOPAL;

- Step 5. Go to Step 2;
- Step 6. Write an error checking test module which determines whether the nominal diagnosis was selected. If it was not selected, then send a message indicating this;
- Step 7. Write an error checking test module which determines whether any fault isolating diagnosis was selected. If none was selected, then send a message indicating this;
- Step 8. Write the failure dictionary in NOPAL;
- Step 9. Write the diagnoses in NOPAL;
- Step 10. Write the failure functions in NOPAL;
- Step 11. Copy DSET08 which contains the ATE function definitions;
- Step 12. Write the test terminals in NOPAL;
- Step 13. [End of NOPAL specification generation] Stop;

The stimulus and measurement functions generated in the NOPAL listing are as shown below. Any additional function definitions written in the NOPAL language may be included in DSET08.

```
FUNCTION : VOLTMETER, TYPE=M, #PINS=2,
          PARAM1=(V,T, LIMIT=(VOLT,+50,-50)),
          PARAM2=(R,S, LIMIT=(OHM,10.0E6,10.0E6)),
          VALUE RETURNED='CONSTANT VOLTAGE' ;
```

```
FUNCTION : AMPMETER, TYPE=M, #PINS=2,
          PARAM1=(I,S, LIMIT=(AMP,5.0,-5.0)),
          PARAM2=(R,S, LIMIT=(OHM,0.01,0.01)),
          VALUE RETURNED='CONSTANT CURRENT' ;
```

```
FUNCTION : OHMMETER, TYPE=M, #PINS=2,
          PARAM1=(R,T, LIMIT=(OHM,10.0E6,0.01)),
          VALUE RETURNED='CONSTANT IMPEDANCE' ;
```

```
FUNCTION : JSUPPLY, TYPE=S, #PINS=2,
          PARAM1=(AMP,T, LIMIT=(AMP,+10.0,-10.0)),
          PARAM2=(R,S, LIMIT=(OHM,0.01,0.01)) ;
```

```
FUNCTION : ESUPPLY, TYPE=S, #PINS=2,
          PARAM1=(V,S, LIMIT=(VOLT,50.0,-50.0)),
          PARAM2=(G,S, LIMIT=(OHM,100.0,100.0)) ;
```

CHAPTER 7 CONCLUSION AND SUGGESTIONS FOR FUTURE RESEARCH

7.1 CONCLUSION

This report describes the research and development of a methodology which automates the process of test design for analog circuits. Three significant aspects of the system are emphasized here:

Man-Machine Interface: The minimum input to the system is the circuit description of the prospective unit under test. This is a simple task which could be performed by a technician having a minimal training in electronics. A proficient user can provide more information and guide the operation of the system. This is illustrated by the many input options described in Appendix A. The options include the ability to define complex failure functions, new models of components, changing of the accuracy in measurements, specifying the acceptable level of fault isolation (thereby controlling the quality of the test program), specifying the needed facilities in the ATE and finally adding test strategies which are evaluated similarly as the system standard tests, including interaction with the ATE operator. Another aspect of the man-machine interface is the variety of reports produced which self document the test design. A variety of error and warning messages alert the user to incompleteness, inconsistencies and ambiguities in the input. The user is also alerted when numerical analysis

problems are encountered during the simulation of the failures.

Knowledge Base: The entire test design and programming methodology has been developed and analyzed prior to the computer implementation. The basic methodology is implemented including two knowledge bases. The electrical engineering knowledge base includes the modelling of catastrophic failures of components, devices, systems, and the generation of candidate test strategies. This knowledge base includes the recognition of the logical implications of conjunctions or disjunctions of the passing or failing outcomes of tests which are used to reduce the ambiguity in fault isolation. The latter has optimization implications which are described below. The computer programming knowledge base includes the generation of test specifications in a nonprocedural language. The information theoretic entropy measure is used as a figure-of-merit to grade the fault isolation capability of each test specification. Tree data structures are used to clarify the operation of the fault isolation algorithm.

Optimization: The main objectives in optimization are: (1) to reduce the number of test setups required, (2) to reduce the number of decision limits which are found from the failure symptoms, and (3) to reduce the number of test modules to be evaluated in order to select a specific diagnosis. The elimination of redundant tests is based on

the concept that there must be useful diagnoses from each test even when it is evaluated in conjunction with other tests, otherwise the test would be deleted. The inclusion of a test in the test specifications always increases the fault isolation capability of the total system. The diagnoses produced by the system are such that the test execution sequencing can be based on a top down concept in testing. The generic tests at the top level are called functional tests. The intent of functional testing is to determine if the UUT is operational or not. If the measurement made on a UUT fails in a top level test, the lower level fault isolation tests are performed.

The system has been written in ANSI FORTRAN to provide easy transportability between different computer systems. It has been successfully used on UNIVAC 70/46, UNIVAC 90/70, and IBM 370/168 computers. The total NOPAL system has been tested with several examples. The computer cost of the example given in Chapter 5 was approximately \$100 at the University of Pennsylvania, David Rittenhaus S/360 Mod 65 computer.

7.2 SUGGESTIONS FOR FUTURE RESEARCH

A software system of this magnitude can never be considered complete. During the course of the research many problems associated with automatic test design were encountered. Some of these problems have been resolved and some of them are proposed here for future research.

1. In this work candidate tests are selected and simulated without apriori estimate on their usefulness to fault isolation capability. Their usefulness is determined after failure simulation. New test strategies involving frequency domain analysis and transient analysis may be investigated. These tests may be selected by an intelligent preprocessor embodied in the test generator program. Before the simulation by a CANA program these candidate tests may be sorted according to a heuristic figure-of-merit to reduce the simulation time.
2. A simple test program generator may be written after the first or second minimization process which will quickly produce a test program in ATLAS, BASIC or OPAL to verify the component and failure modelling on ATE before an optimal program is produced.
3. By implicitly allowing test sequencing through component protection and failure rate index, an incomplete failure symptom table may be used to save simulation time in certain test strategies involving frequency and time

domain analyses. An incomplete failure symptom table does not contain all the failure symptoms of the failures listed in the failure dictionary.

4. The failure simulation approach yields a large amount of information about the behavior of the circuit. The data created in failure simulation can be used for further failure analysis and prediction. For example, after the first failure, the overstressed components may be detected and inserted into the failure dictionary as multiple or cascaded failures. Then, they too may be included in the failure simulation as new failure modes of the UUT. This process may be repeated until a fixpoint in the failure states is reached. The loading factor of each component may be calculated in a manner as described in MIL-HDBK-217B to generate the failure rate indices for each failure automatically.
5. Testing with component protection has not been built in this implementation. The FITS system is capable of generating the data base necessary to detect the overstressing conditions of a UUT. An interesting research area involves the development of a methodology to specify the ambiguity level and to minimize the number of tests while protecting components. This problem is isomorphic to the generalized traveling salesman problem. This problem is agreed upon as an unsolvable problem (in polynomial time) in operations

research and computer science. In the generalized traveling salesman problem the state space consists of countries, cities, rivers and roads. A salesman must visit a number of cities or countries, and given the distance between various pairs of cities, he would like to find the shortest route so that he visits each city or country once and returns to the point of departure. A country is said to have been visited if any city in that country is visited. The rivers act as barriers and require the satisfaction of certain constraints before they can be crossed. The correspondence to the optimization of fault isolation logic is as follows: The cities are the final diagnoses. The countries are intermediate fault isolation results. The rivers are the component protection requirements. The roads are the assertions that can be made. The length of a road is inversely related to the reliability of an assertion. The path the salesman should follow is the diagnosis selection logic.

ACKNOWLEDGEMENT

The author wishes to express his gratitude to Professors Raymond Berkowitz and Noah Prywes for their advice and supervision.

REFERENCES

- [1] R. F. Garzia, "Fault Isolation Computer Methods", NASA Report No. CP-1758, NTIS Accession No. N71-19117, Computer Sciences Corporation, Huntsville, Alabama, February 1971.
- [2] A. M. Greenspan, L. J. Rytter, "Computer Aided Test Generation for Analog Circuits", WESCON - Western Electronics Show and Convention Records, Session 10, San Francisco, California, September 11-14, 1973.
- [3] J. Lustig, D. M. Goodman, "Trends in the Development of Automatic Test Equipment", Project SETE Report 210/106, National Aeronautics and Space Administration, June 1973.
- [4] M. Eleccion, "Automatic Testing: Quality Raiser, Dollar Saver", IEEE Spectrum, August 1974, pp. 38-43.
- [5] M. Eleccion, "Automatic Test Equipment: Hardware and Software", IEEE Spectrum, June 1976, pp. 60-64.
- [6] K. To, R. E. Tullos, "Automatic Test Systems", IEEE Spectrum, September 1974, pp. 44-52.
- [7] C. Tinaztepe, R. Berkowitz, "Automatic Test Program Generation for Automatic Testing Systems", Progress Report, Prepared for U. S. Army, Frankford Arsenal, University of Pennsylvania, Philadelphia, Pennsylvania 19174, March 1976.
- [8] H.Y. Chang, E.G. Manning, G. Metze, "Fault Diagnosis of Digital Systems", Wiley, New York, N.Y., 1970.
- [9] A.D. Friedman, P.R. Menon, "Fault Detection in Digital Circuits", Prentice-Hall, Englewood Cliffs, N.J., 1970.
- [10] F.F. Sellers, M.Y. Hsiao, L.W. Bearson, "Error Detecting Logic", McGraw-Hill, New York, N.Y., 1968.
- [11] S. Lee, "Digital Circuits and Logic Design", Prentice-Hall, Englewood Cliffs, N.J., 1976.
- [12] Special Issue on Fault-Tolerant Computing, IEEE Trans. Computers, Vol. C-20, November 1971.
- [13] Special Issue on Fault-Tolerant Computing, IEEE Trans. Computers, Vol. C-22, March 1973.

- [14] B.B. Gordon, "Survey of Current Fault Isolation Techniques", Proceedings, Seminar on Automatic Checkout Techniques, Battelle Memorial Institute, September 1962.
- [15] "Checking and Fault Isolation", Chapter 7, The Formulation of Automatic Checkout Techniques, Battelle Memorial Institute, March 1962.
- [16] R.F. Garzia, "Fault Isolation Computer Methods", NASA Report No. CP-1758, NTIS Accession No. N71-19117, Computer Sciences Corporation, Huntsville, Alabama, February 1971.
- [17] F. Liguori, Editor, "Automatic Test Equipment: Hardware, Software and Management", IEEE Press, New York, N.Y., 1974.
- [18] M. Eleccion, "Automatic Testing: Quality Raiser, Dollar Saver", IEEE Spectrum, August 1974, pp. 38-43.
- [19] K. To, R.E. Tullos, "Automatic Testing Systems", IEEE Spectrum, September 1974, pp. 44-53.
- [20] M. Eleccion, "Automatic Test Equipment: Hardware and Software", IEEE Spectrum, June 1976, pp. 60-64.
- [21] "Circuits Manufacturing", Benwill Publishing Company, Brookline, Massachusetts 02146.
- [22] S.I. Finkel, R.N. Nilson, E.S.T. Clair, "A Mathematical Automatic Fault Isolation in a Complex System", AD-413305, Vitro Laboratories, May 1963.
- [23] L. Buschbaum, M. Dunning, T.J.B. Hannom, L. Math, "Investigation of Fault Diagnosis by Computational Methods", AD-601204, Remington Rand, UNIVAC, May 1964.
- [24] R.S. Berkowitz, R.L. Wexelbrat, "Statistical Considerations in Element Value Solutions", IRE Transactions on Military Electronics, Vol. MIL-6, No. 3, July 1962, pp. 282-288.
- [25] R.S. Berkowitz, P.B. Krishnaswamy, "Computer Techniques for Solving Electronic Circuits for Fault Isolation", IEEE Transactions on Aerospace, Proceedings of the International Conference and Exhibit on Aerospace Support, Vol. AS-1, No. 2, August 1963, pp. 1080-1099.
- [26] W.L. Stahl, J.H. Maenfaa, "Development of Advanced Fault Diagnosis Techniques", AD-814457, Scully International Inc., May 1967.

- [27] R. Garzia, "Fault Isolation Systems Via Bode Diagram", Proceedings of the Conference on Automated Support Systems for Advanced Maintainability, ASSC Record 1972, pp. 72-101.
- [28] N.N. Puri, C.N. Weygandt, "Transfer Function Tracking of Linear Time Varying by Means of Auxiliary Simple Lag Networks", Joint ACC, 1963.
- [29] J.E. Valstar, "Fault Isolation of Electronic Circuits Using Transfer Function Methods", AD-608176, Air Force Aero Propulsion Laboratory, Wright Patterson A.F.B., Ohio, October 1964.
- [30] E.J. Bayly, U.S. Lenadi, "A Method of Dynamic System Testing", Proceedings of Eighth Conference on Military Electronics, September 1964.
- [31] F.D. Brown, N.F. McAllister, P.R. Perry, "An Application of Inverse Probability to Fault Isolation", IRE Transactions on Military Electronics, Vol. MIL-6, No. 3, July 1962, pp. 260-267.
- [32] J.P. Chorzel, J.R. Thompson, R.G. Myers, "System Parameter Measurement Using Transfer Response Sampling", from Garzia [16].
- [33] P.B. Kraabel, "Power Spectra Analysis as a Means of On-Line Checkout", from Garzia [16].
- [34] R.F. Barry, "Fault Isolation by Parameter Identification", Automatic Support Systems Symposium, RCA Aerospace Systems Division, October 1968.
- [35] R.F. Barry, R.S. Fisher, R.R. Mattison, "Programmed Algorithm for Test Point Selection and Fault Isolation", Vol. 1, AD-487520, Vol. 2, AD-487521, RCA Aerospace Systems Division, August 1966.
- [36] "Dynamic Testing - A Technique for Automatic Checkout of Closed-Loop Systems", Contract NASW410, General Electric Apollo Systems Department, April 1968.
- [37] "The Compiler for the Program Language for Automatic Checkout Equipment (PLACE)", Part 1, Technical Report AFAPL-TR-68-27, Air Force Aero Propulsion Laboratory, Wright Patterson Air Force Base, Ohio 45433, May 1968.
- [38] "Abbreviated Test Language for Avionic Systems", ARINC Specification No. 416-5, Aeronautical Radio, Inc., Annapolis, Maryland, March 1971.

- [39] "Abbreviated Test Language for Avionic Systems", IPG/TS/126, British Aircraft Corp., Herfordshire, England, November 1970.
- [40] T.A. Ellison, "ATLAS - Abbreviated Test Language for Avionic Systems", WESCON Conv. Record, August 1971.
- [41] "Operational Performance Analysis Language - OPAL Definition of, Syntax of and Semantics of", Proposed MIL-STD-1462, Change 2, U.S. Army, Frankford Arsenal, Philadelphia, Pennsylvania 19137, September 1976.
- [42] N.S. Prywes, "Automatic Computer Program Generation for Testing Systems", Report #FCF-3-75, U.S. Army, Frankford Arsenal, Philadelphia, Pennsylvania 19137, January 1975.
- [43] H. Che, Y. Chang, "The NOPAL Language: Specifications and User Manual", Moore School Report 76-04, University of Pennsylvania, Philadelphia, Pennsylvania 19174, August 1976.
- [44] R.L. Mattison, R.T. Mitchell, "UTEC - A Universal Test Equipment Compiler", IEEE Auto. Support Systems Symp. Rec., 12-14 November 1968.
- [45] "HP ATLAS Option for HP9500 ATS", Hewlett-Packard, Palo Alto, California 94304.
- [46] "Program Design Handbook for Automatic Test Equipment", RCA Aerospace Systems Division, Burlington, Massachusetts 01801, June 1968.
- [47] "Military Standardization Handbook, Reliability Prediction of Electronic Equipment", MIL-HDBK-217B superseding MIL-HDBK-217A, Department of Defense, September 1974.
- [48] "Failure Rate Data Book - FARADA", Bureau of Naval Weapons, U.S. Naval Fleet Missile System Analysis and Evaluation Group, Corona, California.
- [49] M. Shooman, "Probabilistic Reliability - An Engineering Approach", McGraw-Hill, New York, N.Y., 1968.
- [50] "EMH Buyers' Guide", Circuits Manufacturing, Benwill Publishing Corporation, Vol. 17, No. 1, January 1977, pp. 50-59.
- [51] "Production Testing Equipment Survey", Circuits Manufacturing, Benwill Corporation, Vol. 14, No. 6, June 1974, pp. 44-57.

- [52] "SCATE MARK-VI Automatic Test Station", General Dynamics, Electronics Division, San Diego, California 92138, September 1973.
- [53] "HP9500 Series Automatic Test Systems", Hewlett-Packard, Palo Alto, California 94304.
- [54] J. Kelly, P.M. Toscano, "EQUATE - New Concepts in Automatic Testing", IEEE Auto. Support Systems Symp. Records, November 1972.
- [55] Frankford Arsenal, "Operational Performance Analysis Language (OPAL), Definition of, Syntax of, Semantics of", Proposed MIL-STD-1462, Change 2, Philadelphia, Pennsylvania 19137, September 1976.
- [56] ARINC, "Abbreviated Test Language for Avionics Systems (ATLAS), Specification 416, Aeronautical Radio, Inc., Annapolis, Maryland 21401, 1972.
- [57] N. S. Prywes, "Automatic Computer Program Generation for Automatic Testing Systems, Report #FCF-3-75, Frankford Arsenal, Philadelphia, Pennsylvania 19137, January 1975.
- [58] H. Che and Y. Chang, "The NOPAL Language: Specifications and User Manual, Moore School Report 76-04, University of Pennsylvania, Philadelphia, Pennsylvania 19174, August 1976.
- [59] Y. Chang, "Automatic Test Program Generation", Dissertation in preparation, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pennsylvania 19174.
- [60] "Military Standardization Handbook MIL-HDBK-105", Department of Defense.
- [61] "SCATE MARK VI, Automatic Test Station", General Dynamics, Electronics Division, San Diego, California 92138, September 1973.
- [62] "HP9500 Series Automatic Test Systems", Hewlett-Packard, Palo Alto, California 94304.
- [63] R. Garzia, "Fault Isolation Methods", NASA Marshall Space Flight Center, Alabama 35812, February 1971.
- [64] "ANP3 and NAP2 Newsletter", Technical University of Denmark, Lyngby, Denmark, December 1976.

- [65] "SUPER*SCEPTRE, Solid-State Model Library", Department of Electrical Engineering, University of South Florida, Tampa, Florida 33620, January 1976.
- [66] "EXTENDED SCEPTRE Program", Air Force Weapons Laboratory, Kirtland Air Force Base, New Mexico, December 1974.
- [67] T. Rubner-Petersen, "Network Analysis Program (NAP2)", Technical University of Denmark, Lyngby, Denmark, March 1973.
- [68] "Military Specification, Control Power Supply, #10559261, MIL-C-14866(CU), Frankford Arsenal, Philadelphia, Pennsylvania 19137, March 1970.
- [69] M. Shooman, "Probabilistic Reliability - An Engineering Approach", McGraw Hill Book Company, New York, New York, 1968.
- [70] A. M. Polovko, "Fundamentals of Reliability Theory", Academic Press, New York, New York, 1968.
- [71] M. F. Goldberg, V. Vaccaro, Editors, "Physics of Failure in Electronics", Spartan Books, Inc., Baltimore, Maryland, 1963.
- [72] H. S. Dodge, "Failure Mechanisms in Semiconductors", M.S. Thesis, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pennsylvania, 1962.
- [73] C. Tinaztepe, R. Berkowitz, "Progress Report, Automatic Test Program Generation for Automatic Testing Systems", Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pennsylvania, March 1976.
- [74] "Military Standardization Handbook, Reliability Prediction of Electronic Equipment", MIL-HDBK-217B, Department of Defense, September 1974.
- [75] "Failure Rate Data Book (FARADA)", Bureau of Naval Weapons, U. S. Naval Fleet Missile Systems Analysis and Evaluation Group, Corona, California.
- [76] F. Linguiti, "A Case Study in FITS Methodology", M. S. Thesis in preparation, Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, Pennsylvania 19174.

- [77] D. Becker, "EXTENDED SCEPTRE User's Manual, AD-A009-594", GTE Sylvania, Inc., Needham Heights, Massachusetts 02194, December 1974.
- [78] L. W. Nagel, "SPICE2 Program", Department of Electrical Engineering and Computer Science, University of California, Berkeley, California 94720, May 1975.
- [79] "NET-2 Program", U. S. Army Material Command, Harry Diamond Laboratories, Washington, D.C. 20438.
- [80] F. F. Sellers, M. Y. Hsiao, L. W. Bearnson, "Analyzing Errors with Boolean Difference", IEEE Trans. Comp., Vol. EC-17, July 1968, pp. 676-683.
- [81] "D-LASAR Program", Digitest Corporation, Dallas, Texas, 75207.
- [82] "TESTAID-III Logic Simulator", Hewlett-Packard, Palo Alto, California 94304, January 1977.
- [83] S. Klein, "Microcomputers on a Chip Come on Strong", Mini-Micro Systems, Vol. 10, No. 2, Modern Data Systems, Hudson, Massachusetts 01749, February 1977, pp.22-26.
- [84] D.B. Armstrong, "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinatorial Logic Nets", IEEE Trans. Electron. Computers, Vol. EC-15, February 1966, pp. 66-73.
- [85] F.F. Sellers, M.Y. Hsiao, L.W. Bearnson, "Analyzing Errors with the Boolean Difference", IEEE Trans. Electron. Computers, Vol. C-17, July 1968, pp. 676-683.
- [86] P.N. Marinos, "Derivation of Minimal Complete Sets of Test-Input Sequences Using Boolean Differences", IEEE Trans. Computers, Vol. C-20, January 1971, pp. 25-32.
- [87] M. Shooman, "Probabilistic Reliability - An Engineering Approach", McGraw-Hill Book Company, New York, N.Y., 1968.
- [88] A.M. Polovko, "Fundamentals of Reliability Theory", Academic Press, New York, N.Y., 1969.
- [89] "Reliability Stress and Failure Rate Data", MIL-HDBK-217 (1962), MIL-HDBK-217A (1965), MIL-HDBK-217B (1974), Department of Defense.

- [90] "Failure Rate Data Book - FARADA", Bureau of Naval Weapons, U.S. Naval Fleet Missile Systems Analysis and Evaluation Group, Corona, California.
- [91] P. Nachman, "Network Statistics for Modelling Computer Aided Network Analysis Programs", M.S. Thesis, The Moore School Electrical Engineering, University of Pennsylvania, Philadelphia, Pennsylvania, May 1974.
- [92] M.F. Goldberg, J. Vaccaro (Editors), "Physics of Failure in Electronics", Spartan Books, Inc., Baltimore, Maryland, 1963.
- [93] H.S. Dodge, "Failure Mechanisms in Semiconductors", M.S. Thesis, The Moore School Electrical Engineering, University of Pennsylvania, Philadelphia, Pennsylvania, May 1962.
- [94] D.A. Calahan, "Computer Aided Network Design", McGraw-Hill, Inc., New York, N.Y., 1972.
- [95] Bell System Technical Journal, Vol. 50, No.4, April 1971.
- [96] W. Feller, "An Introduction to Probability and Its Applications", Wiley, New York, N.Y., 1957.
- [97] "Program Design Handbook for Automatic Test Equipment", RCA Aerospace Systems Division, Burlington, Massachusetts 01801, June 1968.
- [98] D.B. Owen, J.M. Wiesen, "A Method for Computing Bivariate Normal Probabilities", The Bell System Technical Journal, March 1959, pp. 553-572.
- [99] B.J. Karafin, "Optimum Assignment of Component Tolerances for Electrical Networks", BSTJ, Vol. 50, No. 4, April 1971, pp. 1209-1224.
- [100] P.J. Goddard, P.A. Villalaz, R. Spence, "Method for Efficient Computation of the Large Change Sensitivity of Linear Nonreciprocal Networks", Electronic Letters, Vol. 7, No. 4, February 25, 1971, pp. 112-113.
- [101] E.M. Butler, "Large Change Sensitivities for Statistical Design", BSTJ, Vol. 50, No. 4, April 1971, pp. 1225-1242.
- [102] I.A. Cermak, D.B. Kirby, "Nonlinear Circuits and Statistical Design", BSTJ, Vol. 50, No. 4, April 1971, pp. 1173-1195.

- [103] G.B. Dantzig, "Linear Programming and Extensions", Princeton University Press, N.J., 1963.
- [104] D. Becker, "Extended SCEPTRE - Mathematical Formulation", Vol. 2, AD-A009595, GTE Sylvania, Inc., Needham Heights, Massachusetts 02194, June 1974.
- [105] B.D.H. Tellegen, "A Generalized Network Theorem, with Applications", Phillips Res. Rpt., No. 7, 1952, pp. 259-269.
- [106] P. Jessel, C. Tinaztepe, "Techniques for Characterizing CAD Programs", Proc. of 6th Annual Conference on Modelling and Simulation, Pittsburgh, Pennsylvania, Vol. 1, 1975, pp. 243-247.
- [107] J.D. Brule, R.A. Johnson, E.J. Kletsy, "Diagnosis of Equipment Failures", IRE Trans. on Reliability and Quality Control, Vol. RQC-9, April 1969, pp. 23-34.

APPENDIX A

USER'S GUIDE TO FITS

This appendix contains the user's guide to the FITS system. It is organized in two sections: (1) the first section relates to the preparation of input, and (2) the second section relates to the possible job setup configurations in the execution of FITS programs to achieve different affects.

1.1 Preparation of Input for FITS

There are 6 sections of the input to the FITS system. They are: (1) circuit description, (2) test terminals, (3) failure definitions, (4) fault isolation objectives, (5) accuracy specifications, and (6) initial conditions.

Figure A.1 illustrates the general organization of the input file. The sections can be written in any sequence. The beginning of a new section terminates the previous section. End-of-file condition terminates the current input section and the input file. Each input section is described further.


```

CIRCUIT_DESCRIPTION <uut name> <dict name> [<comment>]
.
<circuit description in NAP2 language>
<component> [: <failure> [<function>]] [>] [: <comment>]
.
TEST_TERMINALS <connector> [<comment>]
.
<circuit node> <external node> [<comment>]
.
FAILURE_DEFINITIONS <fda name> [<comment>]
.
DEFINE <id> [<comment>]
.
<modifications to nominal circuit description>
.
END
.
OBJECTIVES <obj name> [<comment>]
<diagnosis %> [<comment>]
<cfi %> <k-ambiguity> [<comment>]
.
ACCURACY <accr name> [<comment>]
<zero discrimination> [<comment>]
<inaccuracy> [<comment>]
<significant digits> [<comment>]
<sort> [<comment>]
<optimize> [<comment>]
<missing> [<comment>]
.
INITIAL_CONDITIONS <init name> [<comment>]
.
BEGIN [DEFINE] [<message>]
<initial conditions in NAP2 language>
(if DEFINE is specified then NOPAL statements are allowed)
END
.
END-INITIAL

```

Figure A.1 General Organization of the Input File

Each section has generally the following form:

```
Column 1
|
| <section identifier keyword> <section name> [<comment>]
|
|           <parameters>
|
| <end-of-section>
|
```

The symbols are delimited by at least one space. Only the first 4 characters of a <section identifier keyword> and the first 8 characters of a <section name> are used for identification.

1.1.1 Circuit Description Section

The circuit description is the most important input to FITS. It is written in the NAP2 language. The semantics of certain statements are extended so that the failure modes of a circuit can be indicated to the FITS system. As far as the circuit analysis program is concerned these extensions are comments.

1.1.1.1 Overview of NAP2 Input

In this section only the overview of the input preparation to the NAP2 circuit analysis program is given, including some examples of input and output of NAP2. The complete documentation can be found in the NAP2 user's manual.

1.1.1.1.1 General Description of NAP2

Nonlinear Analysis Program (NAP2) for electronic circuits can perform steady state analysis, frequency and time domain analyses, noise signal analysis, sensitivity analysis, root sum squared and worst-case tolerance analysis, Fourier analysis of periodic responses, optimization of steady state and transient responses.

The input is entered by a user oriented language. The networks are described by specifying each element, its type, node connections, value and other associated data.

The outputs can be node voltages, element values, voltages, currents, power dissipations, transfer responses, sensitivities, worst-case limits, and other calculated parameters. The outputs can be printed in a tabular form or plotted on a line printer.

There are two versions of the same program. The smaller version requires about 100Kbytes of memory and can solve circuits containing up to 50 nodes and 195 elements. The larger version requires about 250Kbytes of memory and can solve circuits containing up to 500 nodes and 1500 elements.

1.1.1.1.2 Input Organization

An input to the NAP2 program consists of free formatted statements describing the circuit and the type of analysis to be run. The input is organized as follows:

***LIB**

storing of user generated models and subcircuits

***CIRCUIT**

circuit description statements and retrieval of stored models or subcircuits

***(control statements)**

specification of run controls, outputs and types of analysis to be performed

***RUN**

initiates the computation of analysis

***END**

terminates analysis

The ***RUN** command may be followed by modifications to the circuit and run controls. The circuit analysis may then be rerun starting with the initial conditions found in the previous run. The ***END** command terminates the execution of NAP2 and returns the control to the operating system.

1.1.1.1.3 Circuit Elements

Each circuit element must be specified after the ***CIRCUIT** command along with its type prefix and name, node connections, value, and any other associated data. The following element types are supported:

| PREFIX | ELEMENT TYPE |
|--------|----------------------------------|
| R | resistor |
| C | capacitor |
| G | conductor |
| L | inductor |
| E | independent voltage source |
| J | independent current source |
| V | voltage controlled source |
| I | current controlled source |
| M | mutual coupling |
| T | built-in semiconductor reference |
| Q | user defined model reference |
| else | user defined parameters |

The controlled sources may be controlled by any element voltage or current or the time derivative of any element voltage or current. The mutual coupling M may be between any two passive elements R, G, L or C. The type of coupling depends on the elements coupled. All element values, source and coupling values may be constants or linear and nonlinear functions of any named value or circuit response. The named values which may be used are any user defined parameter or one of the following internally stored parameters:

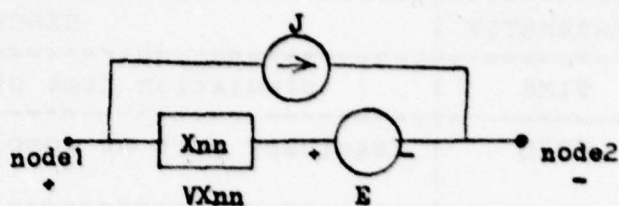
| PARAMETER | DESCRIPTION |
|-----------|---|
| TIME | simulation time of transient analysis |
| FREQ | imaginary part of complex frequency divided by 2π |
| SIGMA | real part of complex frequency |
| CONTROL | parameter used to functionally terminate the run |
| TEMP | ambient temperature |

Node voltages, voltage and current of an RR, L, V, or R=0 type element connected to a node, current of an element that defines a to-branch are primary responses. The transfer function and sensitivity calculations can only be performed on primary responses. The current through any resistor can be made a primary response by prefixing the resistor name by RR.

The reference directions have significance for the sign of the element voltages and currents. Independent voltage sources E and current sources J are specified as subparameters to an element. They follow the conventions in relation to the circuit nodes as shown in Figure A.2.

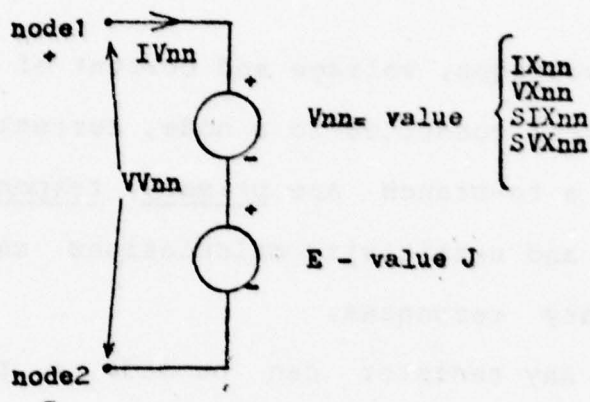
The current is positive if it flows from the first specified node (node1) to the second node (node2). The element currents and voltages include the independent source. This notation results in negative power absorption if independent sources are specified as R or G type elements.

Passive Elements

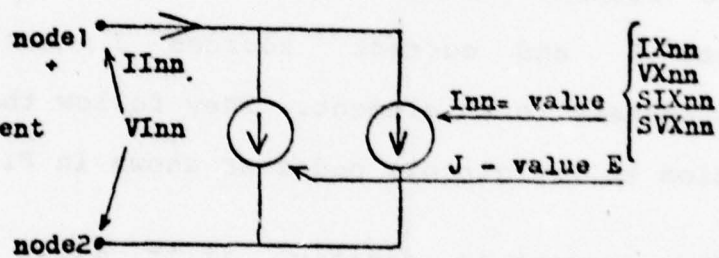


X_n R or G : E or J allowed
 X_n L or C : E or J not allowed

Controlled Voltage Source



Controlled Current Source



E or J are not allowed for S-type couplings.

Figure A.2 Reference Directions

1.1.1.1.4 Function Definition Capability

Functions are entered by means of a function statement giving the name, type, and parameter values of the function. Piecewise linear functions are entered by listing the data points pairwise giving the end points of the linear sections. These tabular functions may be discontinuous at any point, thus step functions are allowed. The ordinate values of the tabular functions may themselves be functions of named values or circuit responses, thus tables may be dynamically used. Functional expressions of the following type can be specified:

$$f(x) = A + B * \text{function}(((x-C)/D)**E) + \text{value1} + \text{value2} + \dots$$

where the "function" may be any one of the Fortran functions ABS, EXP, LOG, SIN, TAN, ATAN or blank. The coefficients A, B, C, D, E and the added values, value1, value2, ... may be constants or computed parameters. The coefficients have default values such that $f(x)=x$. The user is also allowed to write a Fortran subroutine which can be called from the circuit analysis. These functions may be used in accurate modelling of semiconductor devices.

1.1.1.1.5 Built-in Semiconductor Models

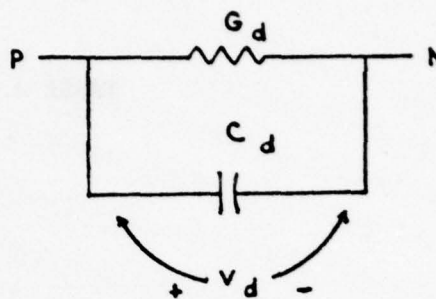
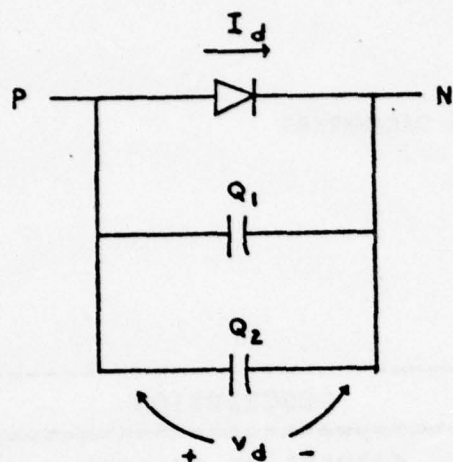
NAP2 has internal models for diodes, bipolar junction transistors and field effect transistors. The user can choose the parameters to model the desired devices. The

diode model is shown in Figure A.3. The user can specify the parameters IS, VT, TT, CJ, FI, GA and GS (see Table A.1). The values of IS and VT are temperature dependent. Temperature dependencies are calculated as follows:

$$\begin{aligned} VT &= VT_0 * (T/298.0) \\ IS &= IS_0 * (T/298.0)^{**3} * \text{EXP}(E_g/VT_0 - E_g/VT) \end{aligned}$$

where $T=273+TEMP$ and $E_g=1.1$ energy gap voltage for silicon. TEMP is the ambient temperature specified by the user. Normally $TEMP=25^{\circ}C$ is used.

The bipolar junction transistor model as shown in Figure A.4 is essentially the Ebers-Moll model where the parameters IS, VT, NI, NV, TF, TR, CE, CC, FI, GA, GZ, NG, AF, AR and GS are specified by the user (see Table A.2). The junction field effect transistor model is shown in Figure A.5. The parameters BE, VP, VU, CS, CD, FI, GA, GZ and NG are specified by the user (see Table A.3). The JFET model is used for MOSFET by choosing the parameters appropriately.



Large-Signal Model

$$I_d = IS(e^{v_d/VT} - 1)$$

$$Q_1 = CJ \int_0^{v_d} \frac{dv}{(1 - \frac{v}{FI})^{GA}}$$

$$Q_2 = TT * I_d$$

Small-Signal Model

$$G_d = \frac{I_d + IS}{VT}$$

$$C_d = TT * G_d + \frac{CJ}{(1 - \frac{v_d}{FI})^{GA}}$$

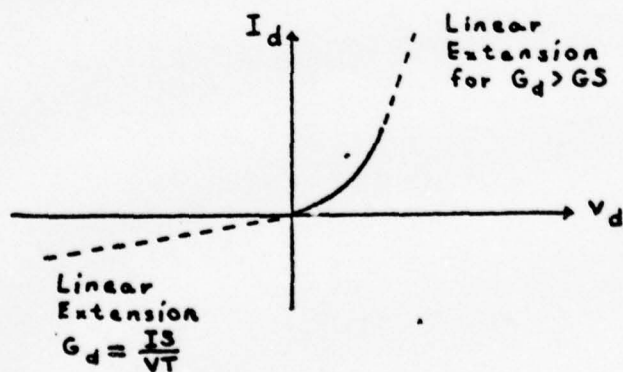
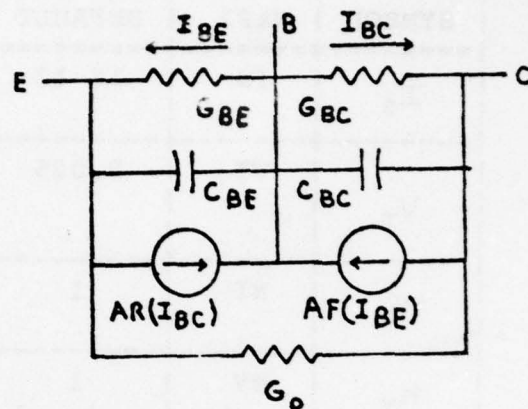
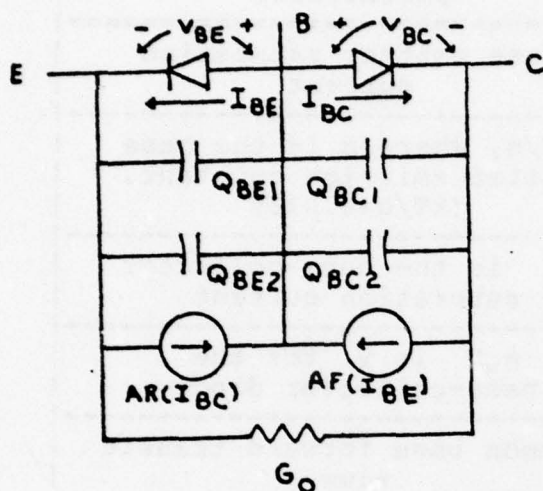


Figure A.3 PN-Junction Diode Models

TABLE A.1 DIODE PARAMETERS

| SYMBOL | NAP2 | DEFAULT | DESCRIPTION |
|----------|------|---------|---|
| I_s | IS | 1E-13 | saturation current |
| V_T | VT | 0.025 | nkT/q , where n is the emission coefficient |
| τ_T | TT | 0 | transit time |
| C_j | CJ | 0 | zero bias junction capacitance |
| ϕ | FI | 1 | junction potential |
| γ | GA | 0.5 | exponent in the capacitance equation |
| G_s | GS | 1 | maximum dynamic conductance |



Large-Signal Model

$$I_{BE} = I_S (e^{v_{BE}/VT} - 1)$$

$$I_{BC} = NI \cdot I_S (e^{v_{EC}/NV(VT)} - 1)$$

$$Q_{BE1} = CE \int_0^{v_{BE}} \frac{dv}{(1 - \frac{v}{FI})} GA$$

$$Q_{BE2} = AF \cdot TF \cdot I_{BE}$$

$$Q_{BC1} = CC \int_0^{v_{BC}} \frac{dv}{(1 - \frac{v}{FI})} GA$$

$$Q_{BC2} = AR \cdot TR \cdot I_{BC}$$

Small-Signal Model

$$G_{BE} = \frac{I_{BE} + I_S}{VT}$$

$$G_{BC} = \frac{I_{BC} + NI(I_S)}{NV(VT)}$$

$$G_O = G_Z + NG(G_{BE} + G_{BC})$$

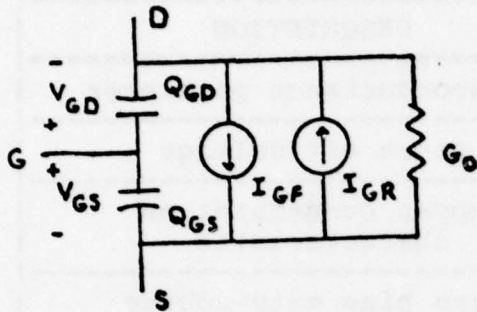
$$C_{BE} = AF \cdot TF \cdot G_{BE} + \frac{CE}{(1 - \frac{v_{BE}}{FI})} GA$$

$$C_{BC} = AR \cdot TR \cdot G_{BC} + \frac{CC}{(1 - \frac{v_{BC}}{FI})} GA$$

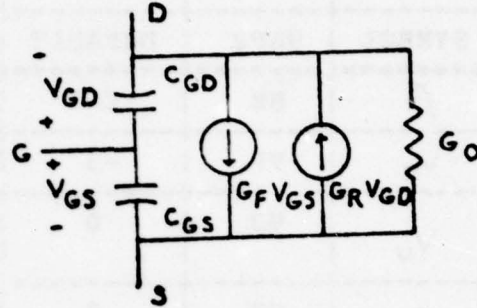
Figure A.4 Bipolar Junction Transistor Models

TABLE A.2 BIPOLAR JUNCTION TRANSISTOR PARAMETERS

| SYMBOL | NAP2 | DEFAULT | DESCRIPTION |
|-------------------|------|---------|--|
| I_s | IS | 1E-13 | base emitter saturation current |
| V_T | VT | 0.025 | nkT/q , where n is the base emitter emission constant. ($kT/q=0.026$) |
| n_I | NI | 1 | $n_I I$ is the base-collector saturation current |
| n_V | NV | 1 | $n_V V$ is V for the base-collector diode |
| $\tau_F \alpha_F$ | TF | 0 | common base forward transit time |
| $\tau_R \alpha_R$ | TR | 0 | common base reverse transit time |
| C_{Ej} | CE | 0 | zero bias base-emitter capacitance |
| C_{Cj} | CC | 0 | zero bias base-collector capacitance |
| φ | FI | 1 | junction potential |
| γ | GA | 0.5 | exponent in capacitance equation |
| G_2 | GZ | 0 | zero bias output conductance |
| n_G | NG | 0 | proportionality factor for output conductance |
| α_F | AF | 0.99 | common base forward current gain |
| α_R | AR | 0.5 | common base reverse current gain |
| G_s | GS | 1 | maximum junction diode conductance |



Large-Signal Model



Small-Signal Model

$$I_{GF} = \begin{cases} 0 & v_{GS} \leq V_P \\ BE(v_{GS} - V_P)^2 & V_P < v_{GS} \leq V_U \\ BE(V_U - V_P)^2 + G_F(v_{GS} - V_U) & v_{GS} > V_U \end{cases}$$

$$G_F = \begin{cases} 0 & v_{GS} \leq V_P \\ 2BE(v_{GS} - V_P) & V_P < v_{GS} \leq V_U \\ 2BE(V_U - V_P) & v_{GS} > V_U \end{cases}$$

$$I_{GR} = \begin{cases} 0 & v_{GD} \leq V_P \\ BE(v_{GD} - V_P)^2 & V_P < v_{GD} \leq V_U \\ BE(V_U - V_P)^2 + G_R(v_{GD} - V_U) & v_{GD} > V_U \end{cases}$$

$$G_R = \begin{cases} 0 & v_{GD} \leq V_P \\ 2BE(v_{GD} - V_P) & V_P < v_{GD} \leq V_U \\ 2BE(V_U - V_P) & v_{GD} > V_U \end{cases}$$

$$Q_{GS} = CS \int_0^{v_{GS}} \frac{dv}{(1 - \frac{v}{F_I})^{GA}} GA$$

$$G_O = G_Z + NG(G_F + G_R)$$

$$Q_{GD} = CD \int_0^{v_{GD}} \frac{dv}{(1 - \frac{v}{F_I})^{GA}} GA$$

$$C_{GS} = \frac{CS}{(1 - \frac{v}{F_I})^{GA}}$$

$$C_{GD} = \frac{CD}{(1 - \frac{v}{F_I})^{GA}}$$

Figure A.5 Junction Field Effect Transistor Models

TABLE A.3 JUNCTION FIELD EFFECT TRANSISTOR PARAMETERS

| SYMBOL | NAP2 | DEFAULT | DESCRIPTION |
|-----------|------|---------|---|
| β | BE | 1E-3 | transconductance parameter |
| V_P | VP | -3 | pinch off voltage |
| V_U | VU | 0 | upper constraint on characteristic |
| C_S | CS | 0 | zero bias gate-source capacitance |
| C_d | CD | | zero bias gate-drain capacitance |
| φ | FI | 1 | gate junction potential |
| γ | GA | 0.5 | exponent in capacitance equation |
| G_z | GZ | 0 | zero drain-source output conductance |
| n_G | NG | 0 | proportionality constant for output conductance |

1.1.1.1.6 Outputs

Any circuit element's value, voltage, current or power dissipation, any node voltage or calculated parameters may be requested for output. The outputs can be saved on selected data sets and printed in a tabular form or plotted on a line printer. NAP2 provides powerful editing features in the output specifications.

1.1.1.1.7 Initial Conditions

Initial conditions for any node voltage or primary current can be set with the *MODIFY command. This command can also be used to change element values, or set element tolerances and sensitivity trackings.

1.1.1.1.8 Run Controls

The *TIME command is used to set the start and stop times for the transient analysis or to vary circuit parameters in the steady state analysis. The time command can also select the interval for outputting data and the type of solution process used - variable step, constant linear step, or logarithmic variations of the step size.

The *FREQ command is used in the frequency domain analysis to set starting and final frequency values. It can also indicate linear or logarithmic variation of the frequency.

The *RUN command passes the control to the numerical analysis section of the program. This command is also used to change any of the 28 internal solution parameters such as minimum and maximum step size, relative truncation error, relative convergence criterium and damping factor for the Newton-Ralphson iteration, order of implicit integration polynomial.

1.1.1.1.9 Types of Analysis

The NAP2 program can perform steady state, frequency domain, and time domain analyses for nonlinear circuits. These analyses are initiated by the commands *DC, *AC, and *TR, respectively. The *DCTR command causes the steady state solution to be calculated first and then the time domain solution is started. Thus the steady state solution can be used as initial conditions in the transient analysis.

Additional analysis capabilities include transfer responses, tolerance analysis, worst-case analysis, Fourier analysis, noise analysis, temperature variation analysis, and automated optimal design. These analysis capabilities are briefly described.

The transfer responses of the form $\partial X_i / \partial J_j$ or $\partial X_i / \partial E_j$ can be calculated where X_i can be any node voltage, element voltage or element current and J_j and E_j can be any current or voltage of a resistor or conductor in the circuit. The transfer responses are requested for output along with the other output requests under the analysis command.

The sensitivity of any primary response X_i with respect to a circuit parameter C_j is calculated by the formula:

$$S_{ij} = \frac{\partial X_i}{\partial X_j} C_j + \sum_{k=1}^n \frac{\partial X_i}{\partial C_k} C_k$$

The summation term represents the contributions due to any n circuit elements which track X_i (i.e., elements which are completely correlated with C_j).

A statistical estimate of the behavior of any primary response X_i can be computed by using the root sum square (RSS) of the weighed sensitivities by the formula:

$$RSS = \sqrt{\sum_{j=1}^n (S_{ij} \Delta tol_j)^2}$$

where Δtol_j is the relative tolerance of circuit parameter C_j and n is the number of circuit elements which have tolerance assigned. The analysis is initiated by a request in the output list options.

The sign of the sensitivities are used to predict the worst cases of the primary responses. Parameters with tolerance specifications are perturbed to their upper or lower tolerance limits depending on the sign of their sensitivity effecting the primary response in the nominal solution. These modified component values are used to approximate to the minimum or maximum worst case of the primary response. *MIN command requests the minimum worst case, *MAX command requests the maximum worst case, and the *WORST command requests both cases to be analyzed.

The sensitivity calculations can also be performed by numerical perturbation of the component values explicitly. The sensitivity becomes equal to the difference between the actual output values and the values from the previous perturbation run. This is accomplished by specifying the SAVEPERT and then PERT options in the *RUN command in two consecutive runs.

Any periodic output quantity can be Fourier analyzed by the *F command. NAP2 calculates the Fourier components and the distortion.

The noise behavior of the network is estimated by the *F command. NAP2 calculates the Fourier components and the distortion.

The noise behavior of the network is estimated by the computation of the transfer responses from all R and G type elements to a primary response X_i .

$$N_i = \sqrt{4kT \Delta \left(\sum_j \left(\frac{\partial x_i}{\partial E_j} \right)^2 |R_j| + \sum_k \left(\frac{\partial x_i}{\partial J_k} \right)^2 |G_k| \right)}$$

The noise analysis applies to the DC analysis only and is initiated by the *NOISE command.

The built-in parameter TEMP represents the ambient temperature. Saturation currents IS and the VT potentials of the PN-junction diodes in the built-in semiconductor models change with TEMP when its value differs from room temperature 25°C.

Automated optimal design is accomplished by adjusting the specified circuit parameters until an optimal solution is found. NAP2 minimizes the root sum square of the relative differences between the calculated and desired values of selected circuit responses by a damped least squares method. The circuit parameters and the constraints that control the optimization are specified in the *OPTIM command.

1.1.1.1.10 Application Example

Input preparation for the NAP2 program is illustrated here by an example showing the steps in preparing input. The sample circuit shown in Figure A.6 is a class-B amplifier intended for low power applications.

The first step in the circuit simulation is to determine the model parameters so that the desired accuracy can be obtained. This is frequently the most difficult phase in computer aided network analysis.

In the sample circuit the nonlinear devices are modelled by the built-in large-signal, Ebers-Moll transistor model and the built-in diode model. These models are shown in Figures A.3 and A.4. The transistor consists of the base-emitter and the base-collector junction diodes with nonlinear capacitors which represent the charge storage effects. The current sources AF and AR represent the common-base forward and reverse current gains. The collector-emitter conductance G_o is made proportional to the dynamic conductances of the junction diodes to improve the steady state simulation of the transistor. The junction diode characteristic is extended linearly when $V_D < 0$, and if the dynamic conductance $G_D > G_S$. The built-in transistor model requires 15 parameters and the diode model requires 7 parameters for a complete definition. The parameters which are not specified take the default values given in Tables A.1 and A.2.

Transistor data handbooks specify the following typical values for the BC107 and BC177 transistors:

| DESCRIPTION | PARAM. | BC107 | BC177 |
|--|-------------|-----------------|-----------------|
| zero bias base-emitter capacitance | CE | 12pF | 24pF |
| zero bias base-collector capacitance | CC | 5pF | 10pF |
| For $I_C=2\text{mA}$, $V_{CE}=5\text{V}$: base-emitter voltage | V_{BE} | 620mV | 650mV |
| transition frequency | f_T | 200MHz | 100MHz |
| common emitter small signal h_e -parameters | h_{ie} | 2.7K | 2.5K |
| | h_{re} | $1.5\text{E}-4$ | $2.3\text{E}-4$ |
| | h_{FE} | 180 | 140 |
| | h_{oe} | $18\text{E}-6$ | $38\text{E}-6$ |
| For $I_C=10\text{mA}$, $I_{Bsat}=0.5\text{mA}$: collector-emitter saturation voltage | V_{CEsat} | 90mV | 75mV |
| base-emitter saturation voltage | V_{BEsat} | 700mV | 700mV |

Most of the data from the handbooks cannot be used directly in the Ebers-Moll model. A simple adjustment is made to evaluate the small-signal equivalent (see Figure A.3) and Ebers-Moll parameters are expressed in terms of the h_e -parameters.

The approximate relationships used are:

| | |
|--|---|
| $h_{ie} = 1/(1-\alpha_F)G_{BC}$ | $G_{BE} = h_{FE}/h_{ie}$ |
| $h_{re} = \frac{1-\alpha_R}{1-\alpha_F} \cdot \frac{G_{BC}}{G_{BE}}$ | $G_{BC} = \frac{1}{1-\alpha_R} \cdot \frac{h_{re}}{h_{ie}}$ |
| $h_{FE} = \alpha_F/1-\alpha_F$ | or $\alpha_F = 1 - \frac{1}{1+h_{FE}}$ |
| $h_{oe} = G_0 + \frac{1-\alpha_R}{1-\alpha_F} G_{BC}$ | $G_0 = \left(\frac{h_{oe} h_{ie}}{h_{FE}} - h_{re} \right) \cdot G_{BE}$ |

The large-signal parameters are then estimated as a best fit from the h-parameters near the operating point: $I_C=2mA$, $V_{CE}=5V$, $V_{BE}=620mV$ for BC107 and $V_{BE}=650mV$ for BC177.

| PARAMETERS | BC107 | BC177 |
|--|--------|--------|
| $AF = 1 - 1/(h_{FE} + 1)$ | 0.9945 | 0.993 |
| AR (estimated) | 0.5 | 0.5 |
| $V_T = I_C \cdot h_{ie}/h_{FE}$ | 30mV | 36mV |
| $IS = I_C \cdot V_{BE}/V_T$ | 2E-12 | 3E-11 |
| $NG = \frac{h_{oe} h_{ie}}{h_{FE}} - h_{re}$ | 1.2E-4 | 4.5E-4 |
| $TF = \alpha_F \tau_F \approx 1/2\pi f_T$ | 0.8ns | 1.6ns |
| $TR = \alpha_R \tau_R \approx 0.5 TF$ | 0.4ns | 0.8ns |
| GA (estimated) | 0.3 | 0.3 |

The reverse transit time τ_R is estimated to be equal to τ_F . In the NAP2 transistor model the reverse biased collector diode conductance G_{BC} is assumed to be a constant equal to I_{CS}/V_{CT} . This gives one equation in the fitting of the collector diode saturation current I_{CS} and the potential V_{CT} . In the saturated region, $I_C/I_{BSat}=20 \ll h_{FE}$; therefore

most of the base current flows in the collector diode. The equations used are:

$$I_{CS} = \frac{1}{1-\alpha_R} I_{B_{sat}} e^{-(V_{BE_{sat}} - V_{CE_{sat}})/V_{CT}}$$

$$G_{BC} = \frac{I_{CS}}{V_{CT}} = \frac{1}{1-\alpha_R} \cdot \frac{h_{re}}{h_{ie}}$$

From these equations I_{CS} and V_{CT} can be found and the model parameters NI and NV become:

| PARAMETERS | BC107 | BC177 |
|-----------------|-------|-------|
| V_{CT} | 55mV | 56mV |
| I_{CS} | 6E-9 | 1E-8 |
| $NV=V_{CT}/V_T$ | 1.63 | 1.56 |
| $NI=I_{CS}/I_S$ | 3000 | 330 |

The large differences in the saturation currents and the V_T potentials of the junction diodes are due to the approximations made and also the limitations of the Ebers-Moll model. The accuracy decreases in the case of large perturbations from the operating point.

The user may define models of semiconductors which are better than Ebers-Moll model for the specific application. When failures in circuits are considered, the linear extensions of the characteristics may result in large errors. More accurate model parameters may be extracted

from laboratory measurements and computer analysis.

For the diode BA100, the following typical values are used: $I_D=1\text{mA}$, and $V_D=635\text{mV}$. If the default value of V_T is assumed the saturation current I_S becomes:

$$I_S = I_D e^{-V_D/V_T} \approx 1.0 \text{E}-14$$

The charge storage effect of the diode is not considered in this example.

When the model parameters are determined, a check of the model behavior can be performed. The test circuit in Figure A.7a is used to trace the transistor's I_C versus V_{CE} characteristic. The input to NAP2 is shown in Figure A.8.

The *TIME statement requests a variation in the the value of $RCC.E$ (i.e., V_{CE}). The variation is performed with variable step size since no step size is specified.

The *PLOT subcommand in the *DC statement requests a plot of the collector current I_{RC} versus the varied collector-emitter voltage $RCC.E$. The plot will consist of 50 lines.

The control parameter HOLD in the *RUN statement keeps the plot output in hold status. Output from more than one run is desired on the same plot. The control parameter OFFL selects the output data set number. The output of the analysis is shown in Figure A.9a. The automatic step control results in many points if the variations of the

response are large and in fewer points if the response is smooth.

The remainder of the input changes the title of the problem by the "*" command. The next analysis is the determination of DC small-signal h_e -parameters. The transfer responses $V_1/JGBB$, $V_1/ERCC$, $IRC/JGBB$ and $IRC/ERCC$ correspond to the h -parameters. $GBB.J$ is a reference to the base current source. $V_1/JGBB$ is a small-signal transfer response which has no relation to the independent sources. The base current is varied from $5\mu A$ to $50\mu A$ and the transfer responses are plotted and printed versus the collector current IRC . The output is shown in Figure A.9b.

The circuit shown in Figure A.7b is used to trace the dynamic junction capacitances CBE and CBC versus the diode reverse voltages. These are varied simultaneously by varying $REE.E$ and $RCC.E$. The model element names have an "&" as a prefix and they are referenced by specifying the transistor name first as a partially qualified name. The results of this analysis is shown in Figure A.9c. The values calculated by NAP2 for these transistors match the data given in the model handbooks.

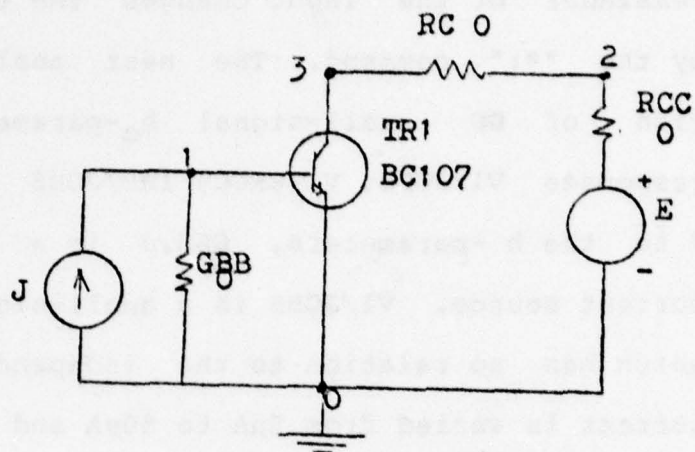


Figure A.7a Test Circuit for IC vs. VCE Characteristics and h_e -parameters

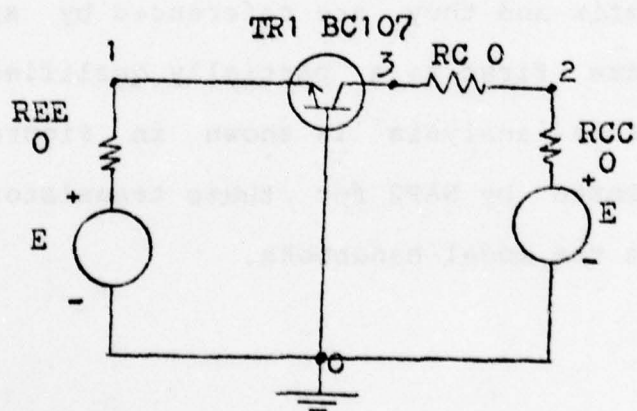


Figure A.7b Test Circuit for Capacitance Characteristics


```

*CIRCUIT
*:      BC107 IC,VCE CHARACTERISTICS
:
:PARAMETER LIST FOR BC107
BC107/NPN/ AF .9945 VT 30MV IS 2.2E-12 NI 3E3 MV 1.63 >
      NG 1.2E-4 TF .8NS TR .4NS CE 12PF CC 5PF GA .3
:BUILT IN TRANSISTOR MODEL, NODES COLL BASE EMIT
TR1 3 1 0 BC107
GBB 0 1 0 J 5UA      = BASE CURRENT SOURCE
RC 2 3 0             = ZERO COLLECTOR RESISTANCE
RCC 2 3 0 E 0V       = VCE VOLTAGE SOURCE
*TIME RCC,E 0 1V     = VARIATION OF VCE FROM 0 TO 1V
*DC *PLOT(50) IRC    = PLOT IBC VERSUS VCE
*RUN      HOLD
*MODIFY GBB,J 1QUA   = MODIFY BASE CURRENT
*RUN      HOLD
*MODIFY GBB,J 2QUA   = MODIFY BASE CURRENT
*RUN      HOLD
*MODIFY GBB,J 5QUA   = MODIFY BASE CURRENT
*RUN      = NOW PLOT
:
*:      BC107 VCE=5V HE-PARAMETERS VERSUS COLL.CURRENT
*TIME GBB,J 5UA 5QUA = VARIATION OF BASE CURRENT
*DC *PLOT(50) IRC) V1/JGBB/ERCC >
      IRC/JGBB/ERCC : PRINT AND PLOT HE-PARAMETERS VERSUS IRC
*RUN
:
*:      BC107 CBE AND CBC DYNAMIC CAPACITANCES VERSUS REVERSE VOLTAGE
GBB-      = DELETE GBB
*TR1 3 0 1 BC107    = CHANGE NODES OF TR1
REE 1 0 0 E 0V       = ADD EMITTER DIODE VOLTAGE SOURCE
*TIME REE,E -.3V 5V > = SIMULTANEOUS VARIATION OF EMITTER AND
      RCC,E -.3V 5V   = COLLECTOR REVERSE VOLTAGE
*DC *PLOT(50) TR1.&CBE> = PLOT DYNAMIC CAPACITANCES
      TR1.&CBC = VERSUS REVERSE VOLTAGE
*RU
*END

```

Figure A.8 Input to NAP2 to Analyze the Circuit Shown in Figure A.7

After sufficiently accurate models are determined, the analysis of the circuit can start. The input description of the circuit shown in Figure A.6 to the NAP2 program is shown in Figure A.10. The transistors have the parameters as determined earlier. The forward current gains are specified on the transistor statements so that they can be included in the sensitivity analysis. The first analysis is the determination of the operating point of the amplifier.

The preferred steady state operating point of this class-B type amplifier is that the quiescent current in the complementary pair TR3 and TR4 should be about 1mA to minimize the cross-over effects. An output voltage at about 3V maximizes the output swing. The parameters that may be changed (R2 and RB3) are specified in the *OPTIM statement. The desired solution (IRE=1mA and V10=3V) is specified in the output list. At the end of the optimization run all circuit responses are printed as shown in Figure A.11a. The adjusted values of R2 and RB3 can be found under the "ALL" listing. Also the AC models corresponding to the Ebers-Moll small-signal equivalent are listed. The optimization stops when 0.1% relative change in the element values can no longer improve the solution. The root sum square of the relative error in the desired solution is also printed.

The next analysis is a worst-case study of the output voltage V10 and the current IRE3. In the first *MODIFY statement several resistors and the voltage source REE.E are assigned the tolerance class 1 with the relative tolerance

+10%. The current gain AF is in tolerance class 2 with positive tolerance 0.2% and negative tolerance 0.8% as specified in the second *MODIFY statement. The nominal solution and component values are lost at the end of a worst-case analysis. If the consecutive worst-case analyses are desired to be more accurate, the nominal circuit can be saved by *SAVE command and *LOADed before each worst-case analysis. The output of the worst-case analysis is shown in Figure A.11b. The output options which are specified before the *WORST subcommand are printed in every worst case as the transfer responses $V1/JR1$ (input resistance), $V10/JR1$ (transfer resistance), and $V10/JG10$ (output resistance). The message "SIGN HAS CHANGED" indicates that the solutions are not true worst cases of $V10$. The sensitivities with the sign change are very small; so their contributions may be ignored. The nominal case of IR3 is based on the actual solution which, in this case, is the maximum case of $V10$. The root sum square of the weighted sensitivities is computed after each worst-case analysis.

The ambient temperature has influence on saturation currents IS and the VT potentials of the junction diodes. When the temperature through the built-in parameter TEMP is varied, the steady state operating point changes. This is shown in Figure A.11c where the temperature dependencies of ITD1, IRE3 and $V10$ are plotted. Above 100 C the temperature dependencies become very strong.

The next analysis is a time domain analysis which is

performed at two different input sine current levels, with amplitude 0.1mA and 0.2mA. The subcommand *F requests a Fourier analysis of V11 to determine the distortion. The time is varied over three periods (3ms). The Fourier analysis is based on the last period when all transient responses have disappeared. The Fourier analysis of more than one period indicates whether this statement is true or not. The results are shown in Figures A.11d and A.11e.

Figure A.11f illustrates the class-B operation of the amplifier with the plots of IRE3 and IRE1.

A small pulse input current excites the small time constants of the amplifier as shown in Figure A.11g. The response of the output voltage V11 to the sharp changes in the input current R1.J is also displayed in this figure.

```

=CIRCUIT
* : DC OPERATING POINT CLASS-B AMPLIFIER
* : PARAMETER LISTS FOR NPN AND PNP TRANSISTORS
BC107/NPN/ AF .9945 VT 30MV IS 2.2E-12 NI 3E3 NV 1.63 >
      NG 1.2E-4 TF .8NS TR .4NS CE 12PF CC 5PF GA .3
BC177/PNP/ AF .993 VT 30MV IS 3.0E-11 NI 330 NV 1.56 >
      NG 4.5E-4 TF 1.6NS TR .5NS CE 24PF CC 10PF GA .3
TR1 2 1 0 BC107 .9945 : BUILT IN TRANSISTOR MODEL
TR2 4 5 12 BC177 .993 : MODES: COLL BASE EMIT
TR3 0 2 9 BC177 .993 : AF IS SPECIFIED LOCALLY
TR4 12 7 8 BC107 .9945 :
BA100/DIODE/ IS 1E-14
TD1 3 2 BA100 : BUILT IN DIODE MODEL
TD2 4 3 BA100 : MODES ANODE CATHODE
: LINEAR ELEMENTS
R1 0 1 3.3K J 0 : RESISTOR WITH INPUT CURRENT SOURCE
R2 10 1 6.8K : INITIAL GUESS FOR R2
R3 5 0 330K
R4 7 2 2K : INITIAL GUESS FOR R4
R5 4 7 200
RE3 10 9 1
RE4 10 8 1
CL 10 11 100UF : OUTPUT CAPACITOR
RL 11 0 8 : LOAD RESISTANCE
REE 12 0 0 E 6V : POWER SUPPLY
: CONTROL STATEMENTS
*OPTIM .001 R2 R3 : OPTIMIZE PARAMETERS
*DC V10=3V IRE3=1MA > : DESIRED SOLUTION
VMA11 ALL : PRINT ALL CIRCUIT RESPONSES
*RUN OFFL 2
:
*RESET
: CONTROL STATEMENTS
*SAVE :SAVE NOMINAL SOLUTION
* : SENSITIVITY AND WORST CASE ANALYSES
GT0 0 10 0 : ADD ZERO CONDUCTANCE AS REFERENCE
*MODIFY 1 .1 R1 R2 R3 > : ASSIGN TOLERANCES
      R4 R5 REE E
*MODIFY 2 .002 .008 TR1.BAF TR2.BAF TR3.BAF TR4.BAF
*DC V1/JR1 V10/JR1/JG10 > : PRINT DC TRANSFER RESPONSES
*WORST V10 *WORST IRE3 : WORST CASE OF V10 AND IRE3
*RUN OFFL 2
:
*LOAD : LOAD NOMINAL SOLUTION
* : TEMPERATURE VARIATION FROM -50 TO 150
*TIME TEMP -50 150 : VARY TEMP FROM -50 TO 150 DEGREES
*DC *PLOT(50) ITR1 IRE3 V10 : PLOT QUIESCENT CURRENTS AND V10
*RUN OFFL 2 : VERSUS TEMPERATURE
:
*LOAD
* : TIME DOMAIN RESPONSES TO SINE WAVE INPUT
SIN/SIN/ : DEFINE SINE WAVE FUNCTION
.R1.J = 1E-4*SIN(2E3PHI*TIME) : CHANGE INPUT CURRENT, FREQUENCY 1KHZ
*TIME 0 3MS : VARY TIME FROM 0 TO 3MS
*DC *PLOT(50) *F V11 > : FOURIER ANALYSIS OF V11
*PLOT(50) *Y IRE3 *Y IRE4 : PLOT EMITTER CURRENTS ON SAME A IS
*RUN OFFL 2 FPER 1MS HOLD : HOLD PLOT OUTPUT
*MODIFY R1.J 2E-4 : CHANGE AMPLITUDE OF SINE CURRENT
*RUN OFFL 2
:

* : TIME DOMAIN RESPONSE TO PULSE INPUT
PULSE/TAB2/ 0 0 60NS 0 > : DEFINE PULSE FUNCTION
      60.1NS 1 360NS 1 360.1NS 0 600NS 0
.R1.J = 2E-4*PULSE(TIME) : CHANGE INPUT CURRENT SOURCE
*TIME 0 600NS : VARY TIME FROM 0 TO 600NS
*DC *PLOT(50) R1.J V11 : PLOT INPUT CURRENT AND OUTPUT VOLTAGE
*RUN OFFL 2
*END

```

Figure A.10 Input to NAP2 to Analyze the Circuit Shown in Figure A.6

***** MAP2 VERSION 760101 DATE 22 2 77 TIME 21 16 20 *****

```
*CIRCUIT
*1 DC OPERATING POINT CLASS-B AMPLIFIER
*1 PARAMETER LISTS FOR NPN AND PNP TRANSISTORS
DC107/PNP/ AF .9945 VT 30MV IS 2.2E-12 NI 3E3 NV 1.63 >
NG 1.2E-4 TF .6NS TR .4NS CE 12PF CC 5PF GA .3
DC177/PNP/ AF .993 VT 30MV IS 3.0E-11 NI 330 NV 1.56 >
NG 6.5E-4 TF 1.6NS TR .6NS CE 24PF CC 10PF GA .3
TR1 2 1 0 DC107 .9945 : BUILT IN TRANSISTOR MODEL
TR2 4 5 12 DC177 .993 : MODELS: COLL BASE LIMIT
TR3 0 2 9 DC177 .993 : AF IS SPECIFIED LOCALLY
TR4 12 7 8 DC107 .9945 :
BAT100/DIODE/ IS 1E-14 : BUILT IN DIODE MODEL
T01 3 2 BAT100 : MODES ANODE CATHODE
T02 4 3 BAT100 :
*LINEAR ELEMENTS
R1 0 1 3.3K J 0 : RESISTOR WITH INPUT CURRENT SOURCE
R2 10 1 6.8K : INITIAL GUESS FOR R2
R02 5 0 330K :
R03 7 2 2K : INITIAL GUESS FOR R03
R04 4 7 200 :
R03 10 9 1 :
R04 10 8 1 :
CL 10 11 100UF : OUTPUT CAPACITOR
RL 11 0 8 : LOAD RESISTANCE
RES 12 0 0 E 6V : POWER SUPPLY
*CONTROL STATEMENTS
*OPTIM .001 R2 R03 : OPTIMIZE PARAMETERS
*DC V10=3V IRES=1MA > : DESIRED SOLUTION
*VALL ALL : PRINT ALL CIRCUIT RESPONSES
*RUN OFFL 6
```

MAP2 OPTIM CYCLE -10 ROOT SUM SQUARE ERROR 4.02E-05

VR 10 3.000014E 00

I RES 1.000040E-03

```
VALL
2 2.376751E 00
1 6.240239E-01
4 3.678270E 00
5 3.345874E 00
12 6.000000E 00
9 2.999014E 00
7 3.604421E 00
8 3.001216E 00
3 3.027510E 00
10 3.000014E 00
11 0.0
-50 -3.598873E-03 I RES
-66 1.000040E-03 I RES
```

| ALL | NODES | VALUE | VOLTAGE | CURRENT | POWER | E SOURCE | J SOURCE |
|------|-------|--------------|---------------|---------------|---------------|--------------|--------------|
| TR1 | 2 0 | 9.509422E-04 | 2.376751E 00 | 2.260153E-05 | 5.371819E-05 | | |
| GB0E | 1 0 | 7.924505E-02 | 6.240239E-01 | 2.377352E-03 | 1.483524E-03 | 6.240239E-01 | 2.377352E-03 |
| GB0E | 1 0 | 7.948890E-11 | 6.240239E-01 | 0.0 | 0.0 | | |
| BAF | 2 1 | 9.945000E-01 | 1.752727E 00 | 2.364276E-03 | 4.143930E-03 | | |
| GB0C | 1 2 | 1.349693E-07 | -1.752727E 00 | -2.365643E-07 | 4.146327E-07 | 0.0 | 0.0 |
| GB0C | 1 2 | 3.690179E-12 | -1.752727E 00 | 0.0 | 0.0 | | |
| BAF | 0 1 | 5.000000E-01 | -6.240239E-01 | -1.182822E-07 | 7.381090E-08 | | |
| TR2 | 12 4 | 2.619044E-05 | 2.321730E 00 | 6.777231E-05 | 1.573490E-04 | | |
| GB0E | 12 5 | 6.486747E-02 | 6.341264E-01 | 2.335229E-03 | 1.527535E-03 | 6.341264E-01 | 2.335229E-03 |
| GB0E | 12 5 | 1.367896E-10 | 6.341264E-01 | 0.0 | 0.0 | | |
| BAF | 5 4 | 9.930000E-01 | 1.667403E 00 | 2.318882E-03 | 1.866975E-03 | | |
| GB0C | 4 5 | 1.762821E-07 | -1.667403E 00 | -2.939685E-07 | 4.902229E-07 | 0.0 | 0.0 |
| GB0C | 4 5 | 7.450267E-12 | -1.667403E 00 | 0.0 | 0.0 | | |
| BAF | 5-12 | 5.000000E-01 | -6.341264E-01 | -1.469843E-07 | 9.614629E-08 | | |
| TR3 | 9 0 | 1.204438E-05 | 2.999014E 00 | 3.612728E-05 | 1.083462E-04 | | |
| GB0E | 9 2 | 2.676957E-02 | 6.222635E-01 | 9.637031E-04 | 5.996772E-04 | 6.222635E-01 | 9.637044E-04 |
| GB0E | 9 2 | 7.497195E-11 | 6.222635E-01 | 0.0 | 0.0 | | |
| BAF | 2 0 | 9.930000E-01 | 2.376751E 00 | 9.569572E-04 | 2.274449E-03 | | |
| GB0C | 0 2 | 1.762821E-07 | -2.376751E 00 | -4.189785E-07 | 9.958073E-07 | 0.0 | 0.0 |
| GB0C | 0 2 | 6.941593E-12 | -2.376751E 00 | 0.0 | 0.0 | | |
| BAF | 2 9 | 5.000000E-01 | -6.222635E-01 | -2.094892E-07 | 1.303575E-07 | | |
| TR4 | 12 8 | 4.750758E-06 | 2.996784E 00 | 1.424650E-05 | 4.272216E-05 | | |
| GB0E | 7 8 | 3.958952E-02 | 6.072045E-01 | 1.187687E-03 | 7.164180E-04 | 6.072044E-01 | 1.187688E-03 |
| GB0E | 7 8 | 4.750436E-11 | 6.072045E-01 | 0.0 | 0.0 | | |
| BAF | 12 7 | 9.945000E-01 | 2.395579E 00 | 1.181155E-03 | 2.829538E-03 | | |
| GB0C | 7 12 | 1.349693E-07 | -2.395579E 00 | -3.233297E-07 | 7.745620E-07 | 0.0 | 0.0 |
| GB0C | 7 12 | 3.464495E-12 | -2.395579E 00 | 0.0 | 0.0 | | |
| BAF | 8 7 | 5.000000E-01 | -6.072045E-01 | -1.616649E-07 | 9.751697E-08 | | |
| TR1 | 3 2 | 8.070802E-02 | 6.507599E-01 | 2.017700E-03 | 1.313038E-03 | 6.507599E-01 | 2.017700E-03 |
| GB0 | 3 2 | 0.0 | 6.507599E-01 | 0.0 | 0.0 | | |
| TR2 | 4 3 | 8.070802E-02 | 6.507599E-01 | 2.017700E-03 | 1.313038E-03 | 6.507599E-01 | 2.017700E-03 |
| GB0 | 4 3 | 0.0 | 6.507599E-01 | 0.0 | 0.0 | | |
| TR1 | 0 1 | 3.300000E 03 | -6.240239E-01 | -1.890981E-04 | 1.708010E-04 | 0.0 | 0.0 |
| R2 | 10 1 | 1.175911E 04 | 2.375990E 00 | 2.020551E-04 | 4.708814E-04 | | |
| R02 | 5 0 | 3.300000E 05 | 3.345874E 00 | 1.619962E-05 | 8.668770E-05 | | |
| R03 | 7 2 | 3.383153E 03 | 1.277670E 00 | 3.626775E-04 | 4.594739E-04 | | |
| R04 | 4 7 | 2.000000E 02 | 7.186963E-02 | 3.692481E-04 | 2.726884E-03 | | |
| R03 | 10 9 | 1.000000E 00 | 1.000040E-03 | 1.000040E-03 | 1.000000E-06 | | |
| R04 | 10 8 | 1.000000E 00 | -1.202095E-03 | -1.202095E-03 | 1.445033E-06 | | |
| CL | 10 11 | 1.000000E-04 | 3.000014E 00 | 0.0 | 0.0 | | |
| RL | 11 0 | 8.000000E 00 | 0.0 | 0.0 | 0.0 | | |
| RES | 12 0 | 0.0 | 6.000000E 00 | -3.598873E-03 | -2.159326E-02 | 6.000000E 00 | 0.0 |

INPUT LIST CPU SECONDS 3.26

Figure A.11a Optimal Design of the Operating Point


```

I
*RESET
I CONTROL STATEMENTS
*SAVE :SAVE NOMINAL SOLUTION
*1 SENSITIVITY AND WORST CASE ANALYSES
*10 0 10 0 : ADD ZERO CONDUCTANCE AS REFERENCE
*MODIFY 1 .1 R1 R2 R3 > : ASSIGN TOLERANCES
R03 R04 R05 E
*MODIFY 2 .002 .001 TR1.SAF TR2.SAF TR3.SAF TR4.SAF
*DC V1/JR1 V10/JR1/J610 > : PRINT DC TRANSFER RESPONSES
*WORST V10 *WORST IAE3 : WORST CASE OF V10 AND IAE3
*RUN OFFL 6

```

```

VN 1/JR1 7.7480970 00
VN 10/JR1 -1.1674750 04
VN 10/J610 8.6061180-01
VN 10 3.0000140 00
NOMINAL VALUE WEIGHTED SENSITIVITY TOLECLASS
SAF 9.9450000-01 -2.7890930-01 2
SAF 9.9300000-01 4.0174240-01 2
SAF 9.9300000-01 -1.1681190-03 2
SAF 9.9450000-01 1.4576100-03 2
R1 3.3000000 03 -4.4153470-01 1
R2 1.1759110 04 4.7182380-01 1
R02 3.3000000 05 -5.5735520-02 1
R03 3.3831530 03 9.7219780-05 1
R04 2.0000000 02 -9.6328020-05 1
REE E 6.0000000 00 6.8692680-02 1
ROOT SUM SQUARE 8.1522180-01

```

```

VN 1/JR1 7.7940370 00
VN 10/JR1 -1.0536450 04
VN 10/J610 1.9628880 00
VN 10 2.3613830 00
MINIMUM VALUE WEIGHTED SENSITIVITY TOLECLASS
SAF 9.9648900-01 -9.5700810-02 2
SAF 9.8505600-01 9.7614110-02 2
SAF 9.9498600-01 -4.4336410-04 2
SAF 9.8654400-01 7.1443030-04 2
R1 3.6300000 03 -3.4528850-01 1
R2 1.0583200 04 3.5182310-01 1
R02 3.6300000 05 -2.8831680-02 1
R03 3.0448380 03 -1.0491280-04 1 SIGN HAS CHANGED
R04 2.2000000 02 5.9540390-05 1 SIGN HAS CHANGED
REE E 5.4000000 00 5.5671690-02 1
ROOT SUM SQUARE 5.1361110-01

```

```

VN 1/JR1 8.0648370 00
VN 10/JR1 -1.2689940 04
VN 10/J610 1.9818420 00
VN 10 4.1354050 00
MINIMUM VALUE WEIGHTED SENSITIVITY TOLECLASS
SAF 9.8654400-01 -5.1885430-01 2
SAF 9.9498600-01 1.6782710 00 2
SAF 9.8505600-01 -2.3922510-03 2
SAF 9.9648900-01 3.2588510-03 2
R1 2.9700000 03 -5.4707720-01 1
R2 1.2935020 04 6.8590330-01 1
R02 2.9700000 05 -1.6748570-01 1
R03 3.0448380 03 9.6492080-04 1 SIGN HAS CHANGED
R04 2.2000000 02 -9.3484300-04 1 SIGN HAS CHANGED
REE E 6.0000000 00 2.0552670-01 1
ROOT SUM SQUARE 1.9813870 00

```

```

I RE3 1.2034250-03
NOMINAL VALUE WEIGHTED SENSITIVITY TOLECLASS
SAF 9.8654400-01 2.1520190-05 2
SAF 9.9498600-01 1.7603370-03 2
SAF 9.8505600-01 9.6616300-08 2
SAF 9.9648900-01 5.7486920-05 2
R1 2.9700000 03 2.2700740-05 1
R2 1.2935020 04 -6.4326860-06 1
R02 2.9700000 05 -1.7567560-04 1
R03 3.0448380 03 3.2265080-04 1
R04 2.2000000 02 -3.0769450-04 1
REE E 6.0000000 00 2.1892600-04 1
ROOT SUM SQUARE 1.6386980-03

```

```

VN 1/JR1 9.7045180 00
VN 10/JR1 -1.2821820 04
VN 10/J610 5.3620790 00
VN 10 3.3360680 00

```

```

I RE3 2.5229930-04
MINIMUM VALUE WEIGHTED SENSITIVITY TOLECLASS
SAF 9.8654400-01 3.4762190-06 2
SAF 9.8505600-01 2.3025050-04 2
SAF 9.8505600-01 2.3662280-08 2
SAF 9.8654400-01 5.9225570-06 2
R1 2.9700000 03 1.5496130-05 1
R2 1.2935020 04 -2.8942420-04 1
R02 3.6300000 05 -6.6449630-05 1
R03 3.0448380 03 9.9084340-05 1
R04 2.2000000 02 -6.9483040-05 1
REE E 5.4000000 00 8.5359610-05 1
ROOT SUM SQUARE 2.8271570-04

```

```

VN 1/JR1 6.3130620 00
VN 10/JR1 -1.0514680 04
VN 10/J610 3.2793590-01
VN 10 2.4621450 00

```

```

I RE3 1.9422740-03
MINIMUM VALUE WEIGHTED SENSITIVITY TOLECLASS
SAF 9.9648900-01 2.4719650-05 2
SAF 9.9498600-01 2.7093290-03 2
SAF 9.9498600-01 6.3298720-08 2
SAF 9.9648900-01 1.0042580-04 2
R1 3.6300000 03 2.0973310-05 1
R2 1.0583200 04 -5.8223510-06 1
R02 2.9700000 05 -2.6862300-04 1
R03 3.7214660 03 3.4753760-06 1
R04 1.8000000 02 -3.3078360-04 1
REE E 6.0000000 00 3.3681750-04 1
ROOT SUM SQUARE 2.7869940-03

```

INPUT LIST CPU SECONDS 3.67

Figure A.11b Sensitivity and Worst-Case Analysis

[illegible]

249

```

1
*LOAD
*1 TIME DOMAIN RESPONSES TO SINE WAVE INPUT
SIN/STN/      1 DEFINE SINE WAVE FUNCTION
*RIJ = 1E-4*SIN(2*PI*TIME) 1 CHANGE INPUT CURRENT, FREQUENCY 1KHZ
*TIME 0 3MS      1 VARY TIME FROM 0 TO 3MS
*OCTR =PLOT(50) =F V11 > 1 FOURIER ANALYSIS OF V11
*PLOT(50) =F IRE3 =F IRE4 1 PLOT Emitter CURRENTS ON SAME A IS
*RUN OFFL 6 &PER TRS HOLD 1 HOLD PLOT OUTPUT

INPUT LIST CPU SECONDS 35.93

*MODIFY RIJ 2E-4      1 CHANGE AMPLITUDE OF SINE CURRENT
*RUN OFFL 6

```

INPUT LIST CPU SECONDS 86.31

HAPZ OCTR TIME DOMAIN RESPONSES TO SINE WAVE INPUT

DATE 22 2 77
TIME 21 16 20

1 TIME
1 VR 11

FOURIER. PERIOD 1.0000E-03 FROM 2.0000E-03 TO 3.0000E-03 INTERVALS 50

| COEFF. | MAGNITUDE | PHASE | EFFECTIVE | SPECTRAL DISTRIBUTION X = 100 |
|--------------------|------------|---------|------------|--|
| 0 | 1.0154E-02 | | | |
| 1 | 1.0338E-02 | -100.34 | 7.3101E-01 | XX |
| 2 | 1.8632E-03 | 70.77 | 1.3175E-03 | XX |
| 3 | 1.8313E-03 | -46.81 | 1.2949E-03 | X |
| 4 | 2.5711E-03 | 84.86 | 1.8181E-03 | XXXX |
| 5 | 1.2020E-03 | 51.16 | 8.4995E-04 | |
| 6 | 1.6767E-04 | 23.92 | 1.1842E-04 | |
| 7 | 3.2742E-04 | 139.53 | 2.3152E-04 | |
| 8 | 8.4290E-04 | 80.32 | 5.9602E-04 | |
| 9 | 1.1460E-03 | -3.61 | 8.1036E-04 | |
| 10 | 2.1532E-04 | 77.56 | 1.5226E-04 | |
| 11 | 6.3488E-04 | 163.51 | 4.4893E-04 | |
| 12 | 8.0643E-04 | 31.81 | 5.7024E-04 | |
| 13 | 2.8318E-04 | 72.94 | 2.0069E-04 | |
| 14 | 1.0400E-03 | 138.02 | 7.3541E-04 | |
| 15 | 1.4926E-03 | 43.39 | 1.0554E-03 | |
| 16 | 2.0663E-03 | 8.96 | 1.4611E-03 | XXX |
| 17 | 9.9710E-04 | -138.51 | 7.0506E-04 | |
| 18 | 1.1636E-03 | -135.49 | 8.2279E-04 | |
| 19 | 1.4597E-03 | 10.35 | 1.0321E-03 | |
| 20 | 8.0174E-04 | 32.18 | 5.6692E-04 | |
| 21 | 5.9851E-04 | 128.80 | 4.2321E-04 | |
| 22 | 1.4573E-04 | 61.47 | 1.0304E-04 | |
| 23 | 9.9545E-04 | -105.61 | 7.0389E-04 | |
| 24 | 3.8439E-04 | 37.14 | 2.7181E-04 | |
| 25 | 1.0985E-03 | -0.00 | 7.7679E-04 | |
| ROOT SUM SQUARE | | | 7.3103E-01 | |
| DISTORTION | | | 5.6107E-03 | |
| EFFECTIVE INTEGRAL | | | 7.3102E-01 | |
| DISTORTION | | | 5.5602E-03 | |
| X | TIME | | | |
| 2 | VR | 11 | | |

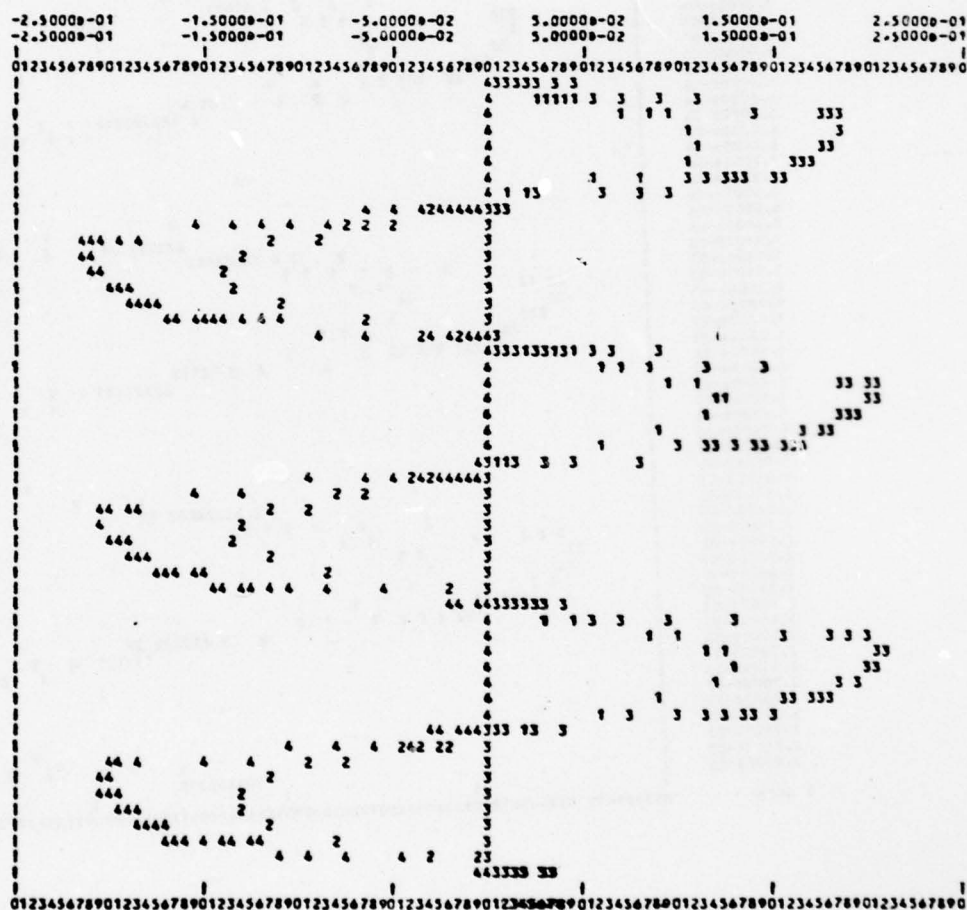
FOURIER. PERIOD 1.0000E-03 FROM 2.0000E-03 TO 3.0000E-03 INTERVALS 50

| COEFF. | MAGNITUDE | PHASE | EFFECTIVE | SPECTRAL DISTRIBUTION X = 100 |
|--------------------|------------|---------|------------|--|
| 0 | 2.0137E-02 | | | |
| 1 | 1.7746E-02 | -99.90 | 1.2550E-01 | XX |
| 2 | 5.4109E-02 | -8.69 | 3.8241E-02 | XX |
| 3 | 1.7834E-01 | -97.44 | 1.2611E-01 | XX |
| 4 | 4.2415E-03 | 105.14 | 2.9992E-03 | XXXX |
| 5 | 4.0748E-02 | 82.79 | 2.8813E-02 | XX |
| 6 | 1.4696E-02 | 164.70 | 1.0391E-02 | XXXXXXXXXXXXXXXXXXXX |
| 7 | 1.8353E-02 | 78.86 | 1.2978E-02 | XXXXXXXXXXXXXXXXXXXX |
| 8 | 1.1709E-03 | 36.83 | 8.2796E-04 | |
| 9 | 7.9334E-03 | -97.41 | 5.6097E-03 | XXXXXXXXXX |
| 10 | 7.8969E-03 | -6.80 | 5.5839E-03 | XXXXXXXXXX |
| 11 | 4.7568E-03 | -102.37 | 3.3636E-03 | XXXXX |
| 12 | 1.0482E-03 | 107.97 | 7.4117E-04 | |
| 13 | 2.7314E-03 | 71.81 | 1.9314E-03 | |
| 14 | 3.8357E-03 | 155.76 | 2.7123E-03 | XXX |
| 15 | 2.5186E-03 | 56.38 | 1.7809E-03 | |
| 16 | 1.0183E-03 | -14.37 | 7.2003E-04 | |
| 17 | 5.9461E-04 | -70.48 | 4.2045E-04 | |
| 18 | 2.6232E-03 | -16.53 | 1.8549E-03 | |
| 19 | 4.5477E-04 | -133.72 | 3.2157E-04 | |
| 20 | 3.9224E-04 | 84.48 | 2.7735E-04 | |
| 21 | 3.8570E-04 | 46.61 | 2.7273E-04 | |
| 22 | 8.4523E-04 | 130.36 | 5.9767E-04 | |
| 23 | 4.8905E-04 | 0.80 | 3.4581E-04 | |
| 24 | 6.7374E-04 | 6.66 | 4.7641E-04 | |
| 25 | 4.6995E-04 | -0.00 | 3.3231E-04 | |
| ROOT SUM SQUARE | | | 1.2624E-01 | |
| DISTORTION | | | 1.0797E-01 | |
| EFFECTIVE INTEGRAL | | | 1.2624E-01 | |
| DISTORTION | | | 1.0797E-01 | |

Figure A.11d Fourier Analysis of Output Voltage V12

DATE 22 2 77
TIME 21 14 20

4413



251

DATE 22 2 77
TIME 21 16 20

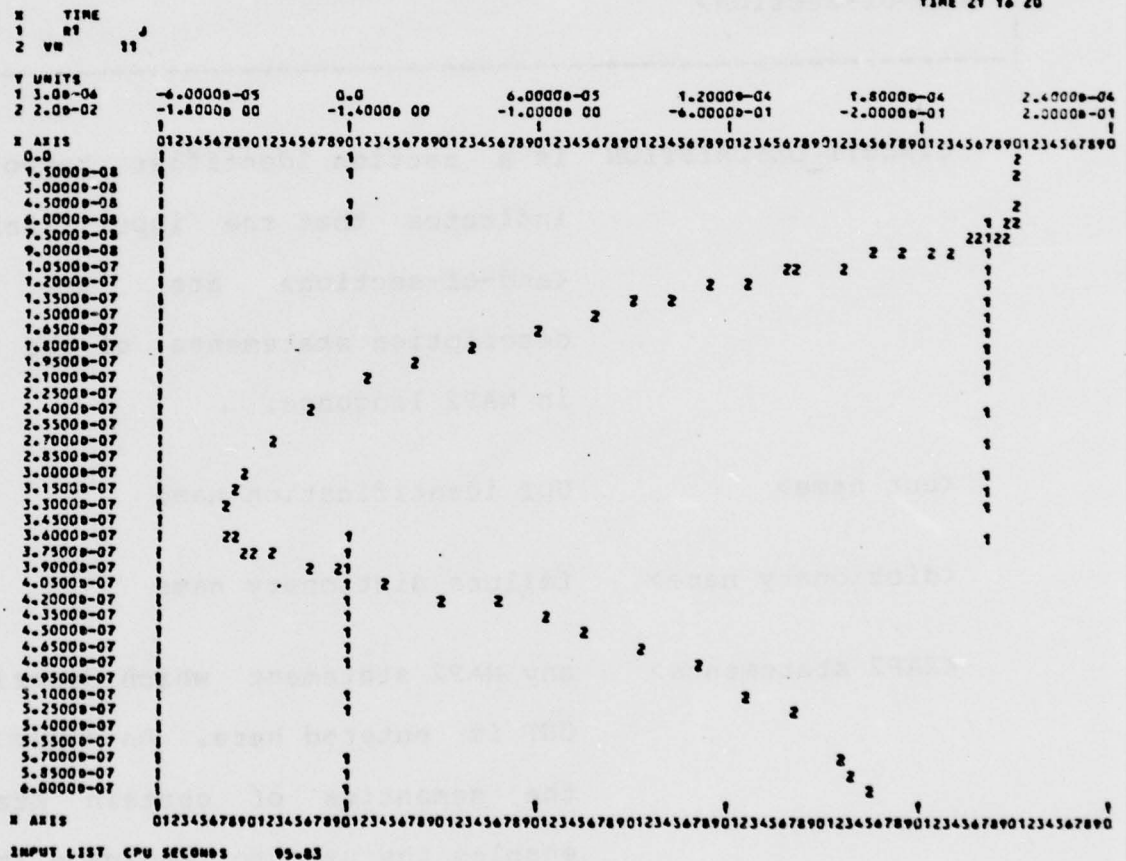


Figure A.11g Time Domain Analysis of Pulse Input

1.1.1.2 Circuit Description Input Section

The circuit description input has the following form:

```
CIRCUIT_DESCRIPTION <uut name> <dictionary name> [<comment>]
.
<NAP2 statements>
<component with failure> [: <failure> [<function>]*] [>]
                                     [ : <comment>]
.
<end-of-section>
```

CIRCUIT_DESCRIPTION is a section identifier keyword which indicates that the input lines until <end-of-section> are the circuit description statements of the circuit in NAP2 language.

<uut name> UUT identification name

<dictionary name> failure dictionary name

<NAP2 statements> any NAP2 statement which describes the UUT is entered here. An extension to the semantics of certain statements enables the user to indicate (to FITS) the components failures which should be automatically included in the failure dictionary. This extension is treated as comment by the NAP2 program.

<component with failure> ::= <component> <node1> <node2>
[<node3>] <value>

<component> name of the component in the circuit

<node1> from or collector node

<node2> to or base node

<node3> emitter node

<value> value, parameter or model name of the
component

FITS can automatically generate failure
dictionary entries if the component
name is prefixed by R, C, L, TD or TQ.

<failure> if the first 4 characters are FAIL,
then this component may have default or
user defined failures.

<function> if DEFAULT is specified, then all
default failures as applicable to the
component type are automatically
generated. Any other name is assumed
to be a user defined failure function
which is provided in the failure
dictionary additions input section. If
nothing follows FAIL, still default
failures are generated. A general
description of the default failures are
listed in Table A.1. The exact form of
the internal and external

representation of the failure definition entry is described in the failure definition input section. The user may change any one of the failure definition parameters consistently.

: indicates that the remainder of the input line contains comments.

> indicates that the input is continued on the following card.

<end-of-section> any section identifier keyword or end-of-file condition terminates the section.

EXAMPLE:

```
RLOAD 5 6 15KOHM : FAILS
```

Resistor RLOAD may fail catastrophically by changing its value to 0.1OHM (short), or to 1.0E9OHM (open).

```
CBIAS 1 0 1UF : FAILURES DEGRADE DEFAULT
```

Capacitor CBIAS may fail by the failure function DEGRADE which is defined in the failure definitions input section. It may also fail catastrophically just like a resistor.

TABLE A.4 CATASTROPHIC FAILURE DEFINITIONS IN FITS

| CIRCUIT ELEMENT | FAILURE FUNCTION | DEFINITION |
|-----------------------------|------------------|--|
| Resistor | OPEN | Replace by a high valued resistor |
| | SHORT | Replace by a small valued resistor |
| Capacitor | OPEN | Replace by a high valued resistor |
| | SHORT | Replace by a small valued resistor |
| Inductor | OPEN | Replace by a high valued resistor |
| | SHORT | Replace by a small valued resistor |
| Diode | OPEN | Replace by a high valued resistor |
| | SHORT | Replace by a small valued resistor |
| Bipolar Junction Transistor | COLL_OPEN | Insert a high valued resistor in series with the collector |
| | BASE_OPEN | Insert a high valued resistor in series with the base |
| | EMIT_OPEN | Insert a high valued resistor in series with the emitter |
| | BC_SHORT | Short base collector with a small resistor |
| | BE_SHORT | Short base emitter with a small resistor |
| | EC_SHORT | Short emitter collector with a resistor |

1.1.2 Test Terminals Input Section

The test terminals input section has the following form:

```
TEST_TERMINALS <connector> [<comment>]
<circuit node> <external node> [<comment>]
.
.
.
<end-of-section>
```

TEST_TERMINALS section identifier keyword indicates that the input lines until <end-of-section> are test terminal names.

<connector> is the connector name of the test terminals. It can be at most 12 characters long.

<circuit node> is the circuit node number of the test terminals in the NAP2 description of the UUT. The first one should be the ground node. It can be at most 3 digits long.

<external node> is the test terminal name of the corresponding circuit node. It can be at most 12 characters long. It is assumed that the test terminal name is also the name of a pin on the connector whose name is given by <connector> .

A circuit may have up to 20 test terminals.

EXAMPLE:

```
TEST TERMINALS EIA_RS_232C COMMUNICATIONS INTERFACE
0 AB SIGNAL GROUND
2 BA DATA -12V to +12V
4 CA GOES HIGH (+12V) WHEN TERMINAL IS READY
8 CF GOES HIGH (+12V) ALLOWS TERMINAL TO RECEIVE
```

This example shows the specification of the test terminals of a UUT. The name of the connector is EIA_RS_232C. Four of the pins on this connector are declared to be test terminals.

1.1.3 Failure Definition Additions Input Section

This section of input allows the user to describe nonstandard failures of a UUT. If the failures of the UUT are associated with the discrete components listed in Table A.4, then an entry in the dictionary is automatically generated. Otherwise, as in the case of multiple or cascaded failures, a new entry in the dictionary must be created by the user. The internal and external (print) organization of the failure dictionary is described in Table A.5.

The failure definitions provided by the user are entered in the following form:

```
-----  
FAILURE_DEFINITIONS <fda-name> [<comment>]  
.  
.  
DEFINE <id> [<comment>]  
.  
  <NAP2 statements>  
.  
END  
.  
.  
<end-of-section>  
-----
```

FAILURE_DEFINITIONS keyword indicates that the following input lines until <end-of-section> are nonstandard failure definitions.

<fda-name> is the failure definitions additions file name

DEFINE keyword indicates that the input lines until the **END** statement are the definitions of the failure whose identification is given next.

<id> the sequence number of the failure as given in the failure dictionary under the column ID. It can be at most 4 digits long.

<NAP2 statements> NAP2 statements which define the failure as modifications to the nominal circuit description. The initial conditions of the circuit can also be included to provide faster convergence in numerical analysis.

END terminates the definition of the failure.

If the nonstandard failures of the UUT can be associated with a component in the circuit, then only the definition of the failure can be given:

EXAMPLE 1:

In the circuit description, the following is declared:

```
R1 1 2 1KOHM : FAILS OPNTOLHI DEGRADED
```

Then, the failure dictionary contains the following definition for failure OPNTOLHI(R1):

```

ID                = 2
COMPONENT NAME    = R1
FAILURE FUNCTION  = DEFINED
NODES             = 1 2
CHANGE           = COMPLEX-FDA
TYPE             = DEFINED
PARAMETER        = SEE --> FDA
VALUE            = 1KOHM
ALIAS            = OPNTOLHI
INDEX            = 0

```

The definition for DEGRADED(R1) would have been identical to the above except:

```

ID                = 3
ALIAS             = DEGRADED

```

The ID's are arbitrarily assigned in this example. If R1 had been a component in a complete circuit description, then the failure ID's might have been different depending on the number of failures preceeding these.

If OPNTOLHI(R1) is a failure which can be represented by a modified resistor with value 1MEG and +200% and -50% tolerances, the definition would be entered as follows:

```

DEFINE 2
      .R1 = 1MEG
      *MODIFY 1 2.0 0.5 R1
END

```

Then, in the worst-case analysis R1 has the minimum value of 0.5MEG and maximum value of 2.0MEG. The tolerance class 1 is arbitrarily chosen. If tolerance class 1 already exists in the nominal circuit description, it is replaced by this new statement. If it does not exist, a new tolerance class 1 is created which stays in effect during the simulation of this failure only. This process may be thought of as a stacking process where the new class is pushed to the top of the tolerance class stack, used in the simulation and then popped up (discarded).

If DEGRADED(R1) is a failure which can be represented by a modified resistor, it can be specified as follows:

```

-----
| DEFINE      3                                     |
|           .R1 = 1.2KOHM                         |
|           *MODIFY 1 0.1 R1                       |
| END                                               |
|-----|

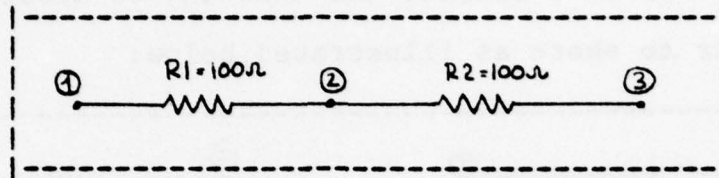
```

This definition indicates that the UUT will have a failure called DEGRADED(R1) if the value of R1 is 1.2KOHM $\pm 10\%$.

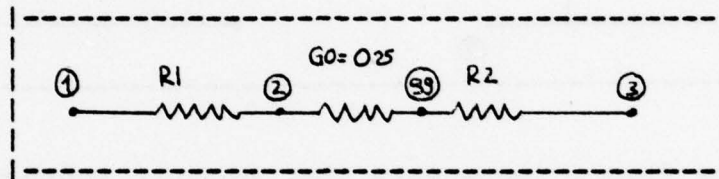
EXAMPLE 2:

A failure of a UUT which is not due to a circuit component failure can also be defined. When a connector strip on a printed circuit board cracks, the failure is not a component failure. Such failures can be modelled by splitting the circuit node which represents the point of connections on the connector strip into two nodes, and then connecting them by a zero conductor.

If the strip connecting R1 and R2 is open due to a crack in the printed circuit board :



it can be represented by :



This failure can be described to FITS by making an entry in the failure dictionary as follows:

```

COMPONENT NAME    = STRIP N
FAILURE FUNCTION  = DEFINED
ALIAS             = PC_CRACK
  
```

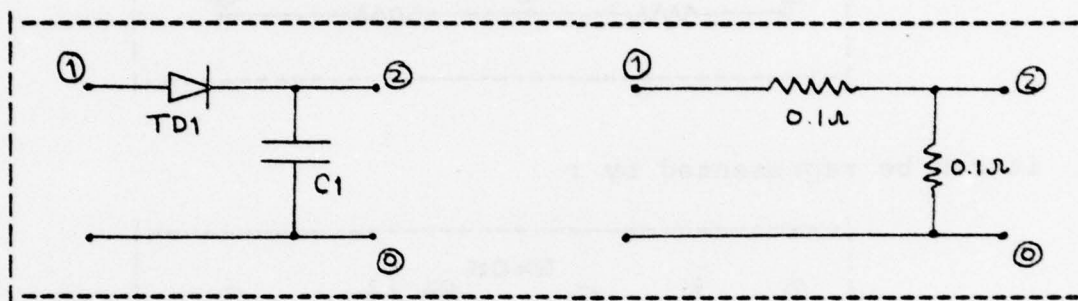
All other entries are ignored. If this failure has ID 2, then the following failure definition can be specified:

```

DEFINE    2
          .R2 99 3 100
          G0 2 99 0
END
  
```

EXAMPLE 3:

Multiple or cascaded failures can be conveniently defined. If in a circuit the shorting of diode TD1 causes a capacitor to short as illustrated below:



The above figure represents the circuit schematic which can be written as the following failure definition:

```

DEFINE      2
      .TD1-      : DROP DIODE
      RDUMMY1 1 2 0.1 : REPLACE BY A RESISTOR
      .C1-       : DROP CAPACITOR
      RDUMMY2 2 0 0.1 : REPLACE BY A RESISTOR
END
  
```

TABLE A.5 OPTIONS IN THE FAILURE DICTIONARY

| NAME | COLUMN RANGE | OPTIONS | DESCRIPTION |
|------------------|--------------|-----------------|---|
| ID. | 1-4 | not applicable | Failure mode sequence number |
| COMPONENT NAME | 5-16 | see own table | Name of the component as it appears in the circuit description |
| FAILURE FUNCTION | 17-20 | see own table | Name of the failure function of the failure of the component |
| NODES | 21-32 | see own table | Identifies the incidence of the component on the circuit nodes |
| CHANGE | 33-36 | see own table | Describes the change in the nominal circuit due to the failure |
| TYPE | 37-40 | see own table | For simple failures identifies the new component type. Otherwise describes where the definition is to be found. |
| PARAMETER | 41-52 | see own table | Value of the new component or subcircuit library member name |
| VALUE | 53-64 | not applicable | Value of the component in the nominal circuit |
| ALIAS | 65-76 | string | Alternate failure function name. This name is passed to the bottom part. |
| INDEX | 82-83 | 2 digit integer | Failure rate index is not currently supported by FITS. If supplied, it is passed on to the bottom part. |

TABLE A.5a OPTIONS IN THE FAILURE DICTIONARY (continued)
(Component Name Options)

| COMPONENT NAME PREFIX | COMPONENT TYPE |
|-----------------------|--|
| R | Resistor |
| C | Capacitor |
| L | Inductor |
| TD | Diode |
| TQ | Bipolar Junction Transistor |
| none of the above | Ignored in automatic dictionary generation but can be inserted to the dictionary manually. |

TABLE A.5b OPTIONS IN THE FAILURE DICTIONARY (continued)
(Failure Function Options)

| INTERNAL | EXTERNAL | DESCRIPTION |
|----------|-----------|-------------------------------|
| 0 | NOMINAL | no failure |
| 1 | OPEN | open |
| 2 | SHORT | short |
| 3 | COLL_OPEN | collector open |
| 4 | BASE_OPEN | base open |
| 5 | EMIT_OPEN | emitter open |
| 6 | BC_SHORT | base collector short |
| 7 | BE_SHORT | base emitter short |
| 8 | EC_SHORT | emitter collector short |
| 9 | BEC_SHORT | all terminals shorted |
| 10 | DEFINED | user defined failure function |

TABLE A.5c OPTIONS IN THE FAILURE DICTIONARY (continued)
(Node Options)

| NUMBER OF ENTRIES | COMPONENT TYPES | ENTRY | INCIDENCE |
|----------------------|---|-------|----------------|
| 2 | Resistor, Capacitor, Inductor, Diode | 1 | from node |
| | | 2 | to node |
| 3 | Transistor | 1 | collector node |
| | | 2 | base node |
| | | 3 | emitter node |

TABLE A.5d OPTIONS IN THE FAILURE DICTIONARY (continued)
(Change Options)

| INTERNAL | EXTERNAL | DESCRIPTION |
|----------|------------------|--|
| -<n> | SAME AS # <n> | Declares the current failure to be identical to failure number <n>. This failure is not be simulated. Its failure symptoms will be identical to failure <n>. |
| 0 | NONE | no change, nominal |
| 1 | TOPOLOGICAL | component type changes |
| 2 | VALUE | only component value changes |
| 3 | LIBRARY | The definition of the failure is in the circuit simulation program model library. |
| 4 | COMPLEX-FDA | User will provide the failure definition in the FAILURE DEFINITION ADDITION file. |

TABLE A.5e OPTIONS IN THE FAILURE DICTIONARY (continued)
(Type Options)

| INTERNAL | EXTERNAL | DESCRIPTION |
|----------|------------------|---------------------------------------|
| 0 | NONE | not used |
| 1 | RESISTOR | failure is represented by a resistor |
| 2 | CONDUCTOR | failure is represented by a conductor |
| 3 | CAPACITOR | failure is represented by a capacitor |
| 4 | LIBRARY MODEL | not used |
| 5 | DEFINED | user defined |

TABLE A.5f OPTIONS IN THE FAILURE DICTIONARY (continued)
(Parameter Options)

| INTERNAL | EXTERNAL | DESCRIPTION |
|----------|---------------------|---|
| <number> | same as internal | Value of the component representing failure. Default values are (OPEN=1.0E9) or (SHORT=0.1). It is filled in according to the catastrophic failure generated. |
| <string> | same as internal | Name of the member in library 3 of NAP2 circuit analysis program. The referenced member should contain a subcircuit description of a failure. |

1.1.4 Fault Isolation Objectives Input Section

This section is written when fault diagnosis and isolation objectives differ from the standard. By default, FITS aims at 85% diagnosis. Any fault isolation level is accepted once 85% diagnosis is reached. It is possible to modify only the diagnosis percentage without requesting any fault isolation. However, FITS still provides the level of fault isolation possible with the tests already available to diagnose the UUT to the desired level.

The fault isolation objectives are written as follows:

```
-----|
| OBJECTIVES <objectives name> [<comment>]|
| <diagnosis percentage> [<comment>]|
| <cfi percentage> <k-ambiguity> [<comment>]|
| .|
| .|
| .|
| <end-of-section>|
|-----|
```

OBJECTIVES keyword indicates that the following input lines until <end-of-section> are fault isolation objectives.

<objectives name> name of the objectives specified

<diagnosis percentage> percent diagnosis desired. It starts on column 1 and is written as a floating point decimal number in 5 columns.

<cfi percentage> cumulative fault isolation percentage.

It is written just like
<diagnosis percentage>.

<k-ambiguity> highest ambiguity level acceptable for
the given percentage of failures. It is
written as a right justified integer on
3 columns after <cfi percentage>.

The fault isolation specifications need not be written if
only diagnosis percentage is to be changed. If they are
specified, then both cumulative fault isolation percentage
and desired ambiguity level should be written in increasing
order. There may be up to 100 specifications.

EXAMPLE:

| OBJECTIVES | STANDARD | (TYPICAL OF INDUSTRY) |
|------------|---------------|-----------------------|
| 85.0% | DIAGNOSIS | |
| 40.0 | 4-AMBIGUOUSLY | |
| 80.0 | 8-AMBIGUOUSLY | |

The specification given above requests that 85% of all the
failures in the failure dictionary should be diagnosed.
Furthermore, 40% of the failures should be isolated in
groups which are 4 or less ambiguous, and 80% of all
failures should be 8 or less ambiguous. The remaining
failures can be isolated to any level once 85% diagnosis is
achieved. The k-ambiguity which is not specified has the
fault isolation percentage level as the next lower ambiguity
class.

1.1.5 Accuracy Specifications Input Section

This section is written when the accuracy specifications differ from the default assignments. The input has the following form:

```
ACCURACY <accuracy name>      [<comment>]
<zero discrimination>          [<comment>]
<inaccuracy>                   [<comment>]
<significant digits>           [<comment>]
<sort>                         [<comment>]
<optimize>                     [<comment>]
<missing>
```

ACCURACY keyword indicates that the following lines until missing belong to the accuracy specifications.

<accuracy name> identifies the specifications

<zero discrimination> see Table A.6

<inaccuracy> see Table A.6

<significant digits> see Table A.6

<sort> see Table A.6

<optimize> see Table A.6

<missing> see Table A.6

All options in this section are written starting from column 1. The floating point numbers are written in 10 columns. The integers are written in 3 columns, right-justified.

TABLE A.6 OPTIONS IN ACCURACY SPECIFICATIONS

| OPTION NAME | TYPE | DEFAULT | VALUE | PURPOSE |
|------------------------|----------|---------|-------|---|
| ZERO DISCRIMINATION | Flt. No. | 1E-6 | | Measurement whose absolute value is less than this value is not included in creating assertions |
| MEASUREMENT INACCURACY | Flt. No. | 0.1% | | Percent inaccuracy in measurements |
| SIGNIFICANT DIGITS | Int. | 4 | | Number of significant digits in the measurements |
| SORT | Int. | 0 | 0 | List the assertions and tests in the order they are created |
| | | | 1 | Sort the assertions within the tests only according to increasing sensitivity |
| | | | 2 | First sort the assertions within the tests and then sort the tests among themselves |
| | | | 3 | Leave the assertions in the order they were created but sort the tests among themselves |
| | | | 4 | Sort the assertions regardless of the tests |
| OPTIMIZE | Int. | 1 | 0 | Include all tests (or assertions) to select diagnoses |
| | | | 1 | Minimize the number of tests (or assertions) to select diagnoses |
| MISSING | Int. | 1 | <n> | Put the missing failure symptoms in the same region as failure <n> |

1.1.6 Initial Conditions Input Section

This section is written when DC strategy tests and other user defined test strategies are to be employed in testing. It also provides a simple means of introducing the initial conditions of the circuit to reduce the number of iterations in the numerical solution of the circuit response. It is observed that in most cases, if the initial conditions of the nominal circuit is close to the final solution, the solution of the circuit with failures is quicker.

The input has the following general form:

```
-----  
INITIAL_CONDITIONS <init. cond. name> [<comment>]  
.  
.  
BEGIN [DEFINE] [<message>]  
.  
.  
[: NOPAL statements if DEFINE is specified]  
.  
.  
<NAP2 statements>  
.  
.  
END  
.  
.  
END-INITIAL  
-----
```

INITIAL_CONDITIONS keyword indicates that the following input lines until END-INITIAL are initial conditions description.

<init. cond. name> initial conditions identification

BEGIN indicates the beginning of a new initial

condition.

DEFINE

when specified it means that this initial condition contains a user defined test strategy and all necessary instructions both in NAP2 and NOPAL languages are provided in this section. When this keyword is not specified it means that only the DC default test strategies should be used, and the user will provide the initial conditions in NAP2 language. The NOPAL statements for the corresponding test are automatically generated.

<message>

if DEFINE is not specified, then the remainder of the card may contain the text of a message to be sent to the operator before the test is conducted.

<NAP2 statements>

NAP2 statements which describe the initial conditions are given here.

END

terminates the current initial condition specification.

END-INITIAL

terminates the initial conditions input.

When no DEFINE keyword is specified, only DC strategy tests are generated. No operator intervention is allowed.

The user is expected to provide the power supplies of the circuit. If so desired, the initial conditions of the circuit which speeds up numerical analysis may also be included.

Two types of power supplies can be specified:

```
-----  
| RE<name> <+node> <-node> <int resis> E <value>  
| GE<name> <+node> <-node> <int condit> J <value>  
|-----
```

The first type of power supply is a voltage source. Positive current flows from the <+node> to the <-node>. <int resis> is the internal series resistance of the voltage source in ohms. <value> gives plus or minus magnitude of the voltage applied.

The second type of power supply is a current source. Positive current flows from <+node> to <-node>. <int condit> is the parallel internal conductance of the current source in mhos. <value> gives the plus or minus magnitude of the current source. FITS uses the second node <-node> as a reference node when making voltage measurements. Therefore, when power supplies have a connecting terminal incident on the ground node, this node should be written as the second node. This is not a restriction but only a notational convenience.

If the power supply specification starts on column 1 of input, then the following NOPAL stimulus statements are

automatically generated:

```
-----  
( < <+external tt> , <-external tt> > VOLT = ESUPPLY  
  ( <value> VOLT , <int resis> OHM ) )  
( < <+external tt> , <-external tt> > AMP = JSUPPLY  
  ( <value> AMP , <int condit> MHO ) )  
-----
```

where <+external tt> is the external test terminal name of the circuit node <+node>, and <-external tt> is the external test terminal name of <-node>. If the corresponding external test terminal name cannot be found in the test terminals input section, it is written as UNDEFINED_<node> where <node> is either <+node> or <-node>.

All other NAP2 statements in this section are copied into the NAP2 candidate tests library. No syntax analysis is performed to check the validity of the specifications.

If the power supply is a voltage source, then NAP2 and NOPAL statements are generated to measure the current through the voltage source. The NOPAL statement generated is:

```
-----  
( < <+external tt> , <-external tt> > AMP = AMPMETER  
  ( I<+external tt>_<no> ) )  
TARGET : I<+external tt>_<no> ;  
-----
```

where the <+external tt> and <-external tt> names are the corresponding test terminal names of the circuit nodes indicated on the voltage supply card. <no> is the sequence

number of the measurement performed.

If the power supply is a current source, then the voltage across it is measured. The NOPAL statement generated is:

```
-----  
| ( < <+external tt> , <-external tt> > VOLT = VOLTMETER  
|   ( V<+external tt>_<no> ) )  
| TARGET : V<+external tt>_<no> ;  
|-----
```

If the target ATE test language has different function names for the above stimuli and measurements, they may easily be modified.

EXAMPLE 1:

```
-----  
| DEFINE      THIS IS A DC TEST  
| RE15V 1 0 1.0 E 15 : 15VOLT POWER SUPPLY  
| *MODIFY V2 2.0 ITQOUT.&GBE 1mA  
| END  
|-----
```

This initial condition specification connects a 15 volt power supply with 1 ohm internal resistance across the circuit nodes 1 and ground. Also the initial condition at node 2 is set to 2 volts, and the bias current of transistor TQOUT is set to 1mA. Thus the DC solution iterations start from these initial conditions for the nominal case and for all failures of the circuit unless they are otherwise modified in the failure definitions.

An extension to the initial conditions specification described above allows the user to control all the NOPAL and NAP2 statements to be generated. FITS, then, simulates the user defined tests for each one of the failures found in the dictionary.

This option is indicated to the FITS system by specifying DEFINE after the BEGIN keyword. Then, any statement which begins with a colon ":" in the first column is considered to be a NOPAL statement and placed in the NOPAL candidate tests library. All other statements are put into the NAP2 candidate tests library. These statements are not checked for syntactic or semantic correctness.

The section relating to NOPAL statements may contain any valid NOPAL statement. FITS automatically fills in TEST name, ASSERTION and LOGIC sections of the corresponding test module if this test is selected to be performed in the final specifications. Furthermore, the STIMULI, MEASUREMENT, CONJUNCTION and MESSAGE keywords are recognized if they are the first symbols in an input card. They are automatically assigned sequence numbers. If a message is to be sent to the operator before the test (operator intervention may or may not be required), then the message text can be given after the MESSAGE keyword. All necessary instructions should be accordingly written to define the message. The command which sends this message to the operator is automatically generated in the "LOGIC" section of the test module when the NOPAL output is being processed.

All other statements which do not start with a colon in the first column are NAP2 statements. In this section, the user has to provide all the instructions to the NAP2 program to simulate the desired test. The failure definitions, test identification numbers, and the "*RUN" commands are automatically inserted. Thus the user can perform DC, AC, or transient analyses and define complex stimulus and measurement specifications.

EXAMPLE 2:

```

-----
BEGIN  DEFINE  A TEST WITH OPERATOR INTERVENTION
:
:  STIMULI :
:  CONJUNCTION :
:    < INPUT , GROUND > VOLT = ESOURCE ( 5 VOLT , 1 OHM ) &
:    < PSIN , GROUND > VOLT = ESOURCE ( 12 VOLT , 1 OHM ) ;
:
:  MEASUREMENT :
:  CONJUNCTION :
:    < OUTPUT , GROUND > VOLT = VOLTMETER ( VOUTPUT ;
:    TARGET : VOUTPUT ;
:
:  DIAGNOSIS SPECIAL : (,,ADJUST,0), ? ;
:  MESSAGE ADJUST :
:    ' TJRN GAIN SWITCH TO POSITION 5 AND TYPE Y ' ;
:
:
:  /* GAIN SWITCH POSITION 5 SETS RGAIN TO 2KOHM */
:  .RGAIN = 2KOHM
:  RESV  1 0 1 E 5 : INPUT SIGNAL
:  REL2V 2 0 1 E 12 : POWER SUPPLY
:  RLOAD 3 0 100KOHM
:  *MODIFY 5 0.2 RGAIN
:  *DC *WORST V3
:
END
-----

```

In this example, a signal source 5 VOLT DC is applied at node 1 (INPUT), and a voltage source 12 VOLT DC is connected to the power supply input node 2 (PSIN). The gain control resistance is changed to 2KOHM (+- 20%) which is on a switch that the operator can control. A voltage measurement is taken at node 3 (OUTPUT). The measurement device loads the output node to the ground with 100KOHMS. The testing takes place after the operator sets the switch to position 5 and replies "Y" to the request. The assertions and fault diagnosis and isolation statements are automatically generated using FITS methodology.

AD-A056 660

MOORE SCHOOL OF ELECTRICAL ENGINEERING PHILADELPHIA P--ETC F/6 9/3
AUTOMATED TEST DESIGN.(U)

JUN 78 C TINAZTEPE

DAAA25-75-C-0650

UNCLASSIFIED

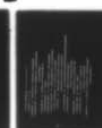
77-03

ECOM-75-0650-F-2

NL

4 of 4

AD
A056 660



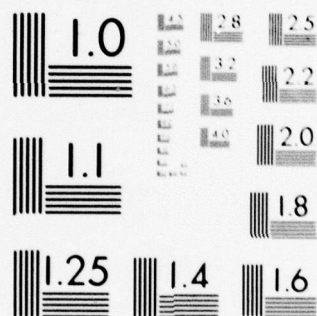
END

DATE

FILMED

9-78

DDC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

1.2 Job Setup Configurations

There are several ways of configuring the program execution sequence in FITS to generate test specifications to the desired effect. The simplest configuration is described with the aid of Figure A.12. The job control cards to set up this configuration is provided in APPENDIX C.

The process starts after the user prepares the input manually. This input is given to the file initialization program (P1) which generates the necessary internal files from the user input. Then, a failure dictionary is generated (P2) and printed (P3). All candidate tests using CC, DC or user defined test strategies are placed into proper files (P4). The candidate tests written in the NAP2 language are analyzed in the circuit analysis program NAP2 (P5). The output of the failure analysis is scanned by the symptom generator program, and a failure symptom table is prepared (P6). Then this table is evaluated by the decision limits finder program (P7) to generate assertions and diagnoses. The decision table is also printed. Then the ambiguity analysis program determines the equivalent failure groups and indicates if the fault isolation objectives are satisfied (P8). The user either chooses to try more tests (P9), or continues to optimization. If more tests are to be tried, there should be some candidate tests remaining (P10). If no more tests are available, then it is not possible to improve on the fault isolation capability. However, if there are some more tests, they are evaluated starting from P4. After the optimization processes (P11), the test specifications are produced in the NOPAL language (P12).

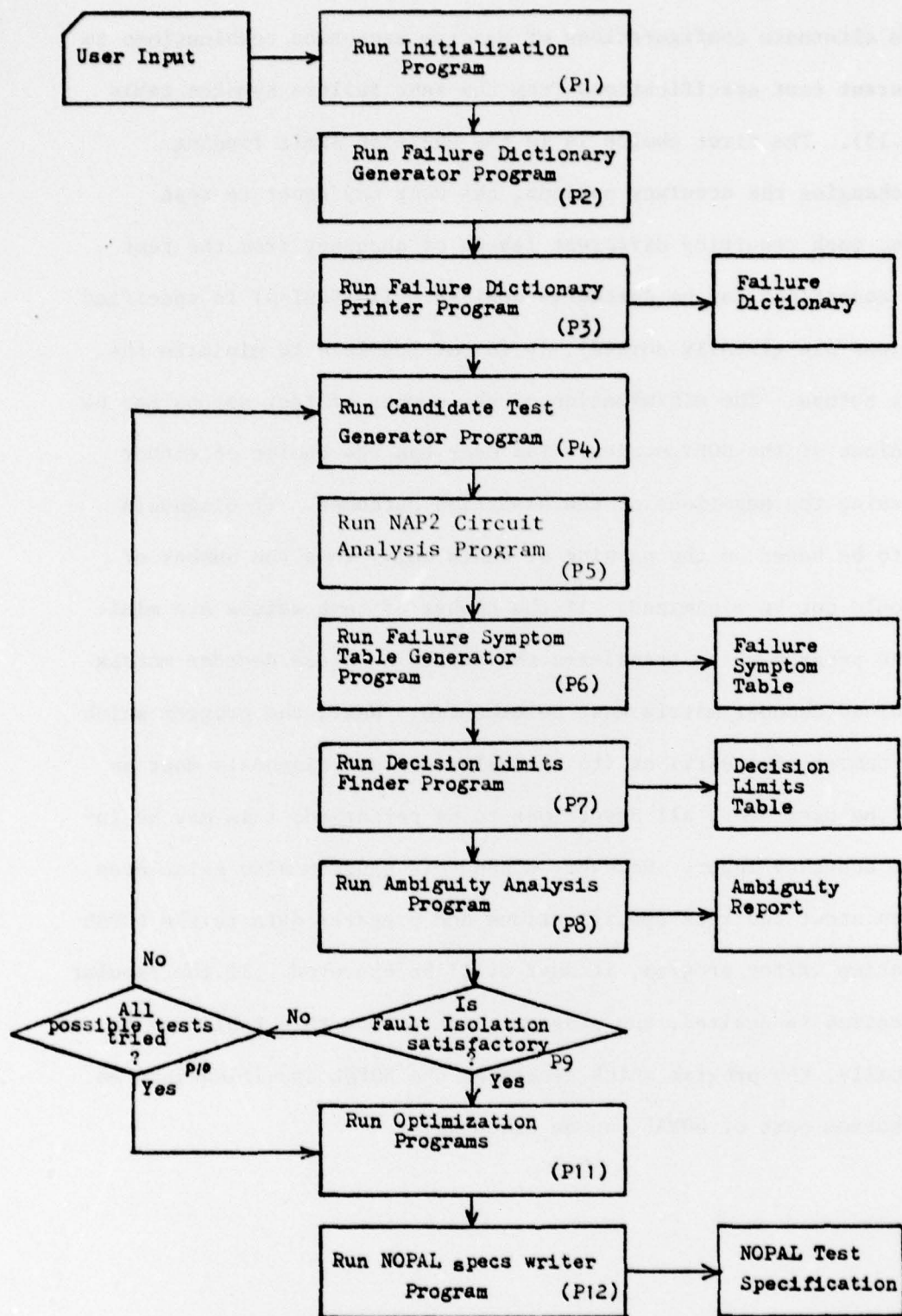


Figure A.12 Simple Run Configuration

There are alternate configurations of program execution combinations to generate different test specifications from the same failure symptom table (see Figure A.13). The first choice is in the decision limit finding process. By changing the accuracy options, the user may generate test specifications, each requiring different levels of accuracy from the test devices. The assertions may be sorted as desired. If (SORT=4) is specified (i.e., assertions are globally sorted), it is not possible to minimize the number of test setups. The minimization of the number of test setups may be omitted regardless of the SORT option. The user has the choice of either using or not using the negations of the assertion outcomes. If diagnosis selection is to be based on the passing of tests only, then the number of assertions should not be minimized. If the number of test setups are minimized, then the program which transforms the multiple valued decoder matrix to a binary valued decoder matrix must be executed. Next, the program which minimizes the number of assertions (to be evaluated) per diagnosis must be executed. If the user wants all assertions to be performed, this may be indicated in the accuracy input. However, since this program also calculates some statistics about the test specifications and prepares data to the NOPAL test specification writer program, it must still be executed. If the tabular NOPAL specification is desired, the program which prints this table may be executed. Finally, the program which generates the NOPAL specifications as input to the bottom part of NOPAL can be executed.

From Failure Symptom Generation

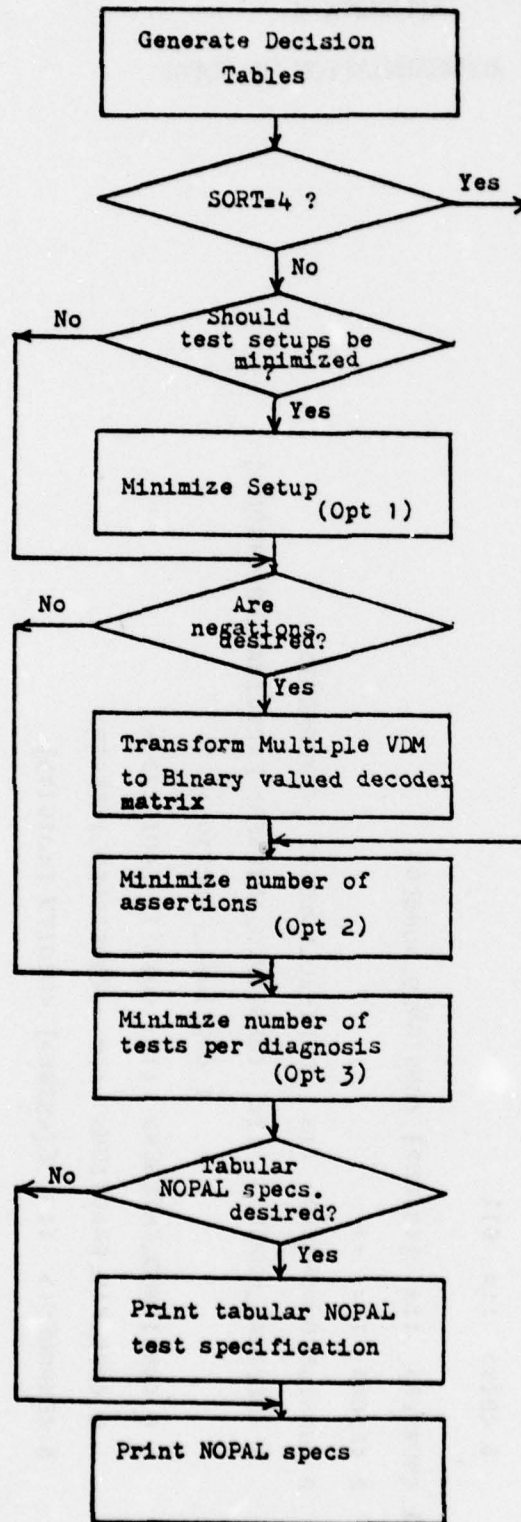


Figure A.13 Alternate Configurations to Generate NOPAL Specifications

APPENDIX B

EBNF REPRESENTATION OF NOPAL

```

1 <STRING_CONST> ::= <CHAR_STRING> | <BIT_STRING>
2 <CHAR_STRING> ::= '[' <FULL_CHAR> ] *
3 <FULL_CHAR> ::= <LETTER> | <DIGIT> | <SPECIAL_CHAR>
4 <LETTER> ::= A B C D E F G H I J K L M N O P Q R S T U V W X Y Z I A I S I _
4 <DIGIT> ::= 0 1 2 3 4 5 6 7 8 9
4 <SPECIAL_CHAR> ::= . | ( | + | ! | & | % | * | | | / | \ | _ | > | ? | : | # | @ | ' | = | " | B L A N K
2 <BIT_STRING> ::= ' <BIT> [ <BIT> ] * ' B
3 <BIT> ::= 0 1 1

1 <NUMBER> ::= [ <SIGN> ] <UNSIGNED_NUMBER>
2 <SIGN> ::= + | -
2 <UNSIGNED_NUMBER> ::= <DECIMAL_NUMBER> [ <EXPONENT> ]
3 <DECIMAL_NUMBER> ::= <UNSIGNED_INTEGER> [ <DECIMAL_FRACTION> ]
    1 <DECIMAL_FRACTION>
4 <UNSIGNED_INTEGER> ::= <DIGIT> [ <DIGIT> ] *
4 <DECIMAL_FRACTION> ::= . <UNSIGNED_INTEGER>
3 <EXPONENT> ::= E [ <SIGN> ] <DIGIT> [ <DIGIT> ]

1 <INTEGER> ::= [ <SIGN> ] <UNSIGNED_INTEGER>

1 <VARIABLE> ::= <IDENTIFIER> [ ( <SUBSCRIPT_LIST> ) ]
2 <IDENTIFIER> ::= <LETTER> [ <TAIL> ] *

```



```

3 <TAIL> ::= <LETTER> | <DIGIT>
2 <SUBSCRIPT_LIST> ::= <ARITH_EXPR> [ , <ARITH_EXPR> ] *
1 <ARITH_EXPR> ::= [ <SIGN> ] <TERM> [ <ADD_OP> <TERM> ] *
2 <TERM> ::= <FACTOR> [ <MULT_OP> <FACTOR> ] *
3 <FACTOR> ::= <PRIMARY> [ ** <PRIMARY> ] *
4 <PRIMARY> ::= <UNSIGNED_NUMBER> | <VARIABLE> | <FUNCTION_CALL>
      | ( <ARITH_EXPR> )
5 <FUNCTION_CALL> ::= <FUNCTION_ID> [ ( <ARGUMENT> [ , <ARGUMENT> ] * ) ]
6 <FUNCTION_ID> ::= <IDENTIFIER>
6 <ARGUMENT> ::= <ARITH_EXPR> | <STRING_CONST>
3 <MULT_OP> ::= * | /
2 <ADD_OP> ::= + | -
1 <COMMENT> ::= /* [<FULL_CHAR>*] */

1 <IF_CLAUSE> ::= IF <BOOLEAN_TERM> THEN
2   <BOOLEAN_TERM> ::= <BOOLEAN_FACTOR> [ V <BOOLEAN_FACTOR> ] *
3   <BOOLEAN_FACTOR> ::= <BOOLEAN_PRIMARY> [ & <BOOLEAN_PRIMARY> ] *
4   <BOOLEAN_PRIMARY> ::= <RELATIONAL_EXPR> | ~ (<BOOLEAN_TERM>)
5   <RELATIONAL_EXPR> ::= <ARITH_EXPR> <RELATION> <ARITH_EXPR>
6   <RELATION> ::= = | > | < | >= | <= | <> | <-> | <-> | <->

```

```

1 <CONN_DIM_EX> ::= <CONNECTOR> [<DIMENSION>]
2 <CONNECTOR> ::= <CONNECTOR_ID> | <CONNECTOR_ID> [<CONNECTOR_ID>]* >
3 <CONNECTOR_ID> ::= <UUT_POINT_ID>
2 <DIMENSION> ::= HZIKHZIMHZIGZIVIKVIMVUUVIAIUAIWIMWUWIOHMIKOHMIOHMIHOHM
IDBIOBMIPCI%IPPSIKPPSIPPSIDEGIDEGIDEGFI<TIME_DIMENSION>
3 <TIME_DIMENSION> ::= HRSIMINISECIMSECIUSECINSECIPSEC
1 <VAL_DIM_EX> ::= <ARITH_EXPR> [<DIMENSION>]
1 <FUNC_DIM_EX> ::= <FUNC_TERM> [<ADD_OP> <FUNC_TERM>]*
2 <FUNC_TERM> ::= <FUNC_FACTOR> [<MULT_OP> <FUNC_FACTOR>]*
3 <FUNC_FACTOR> ::= [<FUNC_MODIFIER> <MULT_OP>] <FUNC_PRIMARY>
4 <FUNC_MODIFIER> ::= <UNSIGNED_NUMBER>
4 <FUNC_PRIMARY> ::= <FUNCTION_ID> [( <FUNC_ARG> [<FUNC_ARG>]* ) ]
| ( <FUNC_DIM_EX> )
5 <FUNC_ARG> ::= [<RELATION>] <VAL_DIM_EX> | [=] <RANGE> | <STRING_CONST> | *
6 <RANGE> ::= <VAL_DIM_EX> +- <VAL_DIM_EX> | <VAL_DIM_EX> +- <ARITH_EXPR> [x]

```

```

1 <NOPAL_SPECIFICATION> ::= [NOPAL] SPECIFICATION [ <SPEC_NAME> ] !
    [ <NOPAL_STMTS> ] *
    END [ <SPEC_NAME> ] !

2 <SPEC_NAME> ::= <LABEL>

3 <LABEL> ::= <IDENTIFIER> ! <UNSIGNED_INTEGER>

2 <NOPAL_STMTS> ::= <TEST_MODULE_SPEC> ! <UNIT_SPEC> ! <ATE_SPEC>

3 <TEST_MODULE_SPEC> ::= <TEST_STEP>

    ! <DIAGNOSIS_DEFINITION> [ <DIAGNOSIS_DEFINITION> ] *
    ! <MESSAGE_DEFINITION> [ <MESSAGE_DEFINITION> ] *

4 <TEST_STEP> ::= TEST [ <TEST_LABEL> ] !

    ! STIMULI <STIM_ID> [ : <BACK_REFERENCE> [ <DECLARATION> ] * ] !
    ! MEASUREMENT <MEAS_ID> [ : <BACK_REFERENCE> [ <DECLARATION> ] * ] !
    ! LOGIC <LOGIC_ID> [ : ] <LOGIC_DIAG_LIST> !
    ! <WAVEFORMS>

5 <TEST_LABEL> ::= <LABEL>

5 <STIM_ID> ::= [ <LABEL> ] [ , <TEST_LABEL> ]

5 <MEAS_ID> ::= <STIM_ID>

5 <LOGIC_ID> ::= <STIM_ID>

5 <LOGIC_DIAG_LIST> ::= <LOGOP_DIAGLBL> [ , <LOGOP_DIAGLBL> ] *

6 <LOGOP_DIAGLBL> ::= <LOGICAL_OPERATOR> <DIAG_LABEL>

7 <LOGICAL_OPERATOR> ::= <LOGICAL_CONNECTIVE> [ <AFTER> ] ! <AFTER>

```



```

8 <LOGICAL_CONNECTIVE> ::= V | & | V~ | &~ | *
8 <AFTER> ::= A | A~

5 <WAVEFORMS> ::= <CONJUNCTION> | <ASSERTION> [<ASSERTION>]*
6 <CONJUNCTION> ::= CONJUNCTION <WAVEFORM_ID> : <CONJUNCTION_BODY>
    [<DECLARATION>]* ;

7 <WAVEFORM_ID> ::= [<LABEL>] [( <STIM_MEAS_LABEL> )]
8 <STIM_MEAS_LABEL> ::= <LABEL>
7 <CONJUNCTION_BODY> ::= <TRIPLET_CONJUNCT> | <BACK_REFERENCE>
8 <TRIPLET_CONJUNCT> ::= <SIMPLE_CONJUNCTION> | <IF_CONJUNCTION>
9 <SIMPLE_CONJUNCTION> ::= <TRIPLET> [& <TRIPLET>]*
10 <TRIPLET> ::= ( <CONN_DIM_EX> <RELATION> <FUNC_DIM_EX> )
    | <CONN_DIM_EX> <RELATION> <FUNC_DIM_EX>
9 <IF_CONJUNCTION> ::= <IF_CLAUSE> <SIMPLE_CONJUNCTION> [ ELSE <TRIPLET_CONJUNCT> ]
8 <BACK_REFERENCE> ::= [ SAME ] AS <STIM_MEAS_LABEL>
    [ EXCEPT <SIMPLE_CONJUNCTION> ]

7 <DECLARATION> ::= <VARIABLE_TYPE> [ : ] <VARIABLE_LIST>
8 <VARIABLE_TYPE> ::= SOURCE | TARGET
8 <VARIABLE_LIST> ::= ( <VARIABLE> [ , <VARIABLE> ] ) * .
    | <VARIABLE> [ , <VARIABLE> ] *
6 <ASSERTION> ::= ASSERTION <WAVEFORM_ID> : <ASSERTION_BODY>
    [<DECLARATION>]* ;

```

```

7 <ASSERTION_BODY> ::= <SIMPLE_ASSERTION> | <IF_ASSERTION>
8 <SIMPLE_ASSERTION> ::= <RELATIONAL_EXPR>
    | <ARITH_EXPR> = <ARITH_EXPR> +- <ARITH_EXPR> [*]
8 <IF_ASSERTION> ::= <IF_CLAUSE> <SIMPLE_ASSERTION> [ ELSE <ASSERTION_BODY> ]
4 <DIAGNOSIS_DEFINITION> ::= DIAGNOSIS <DIAG_LABEL> [:] <DIAG_BODY> |
5 <DIAG_LABEL> ::= <LABEL>
5 <DIAG_BODY> ::= <POSITIONAL_DIAG> | <KEYWORDED_DIAG>
6 <POSITIONAL_DIAG> ::= [<OPERATOR_MESSAGE>] [<OPERATOR_RESPONSE>]
7 <OPERATOR_MESSAGE> ::= ([<AFFECTED_COMPONENTS>][<OTHER_PARAMETERS>]
    [,<TYPE>] [<TIMING>]])
    | <AFFECTED_COMPONENTS>
8 <AFFECTED_COMPONENTS> ::= <COMPONENT_CONJUNCT> [& <COMPONENT_CONJUNCT>]*
    | <COMPONENT_DISJUNCT> [V <COMPONENT_DISJUNCT>]*
9 <COMPONENT_CONJUNCT> ::= <COMP_FAIL_SEQ> | <COMPONENT>
    | <FAILURE_FUNCTION> ( <COMPONENT> [& <COMPONENT>]* )
10 <COMPONENT> ::= <IDENTIFIER>
9 <COMPONENT_DISJUNCT> ::= <COMP_FAIL_SEQ> | <COMPONENT>
    | <FAILURE_FUNCTION> ( <COMPONENT> [V <COMPONENT>]* )
8 <OTHER_PARAMETERS> ::= ( <MSG_ARGUMENT> [<MSG_ARGUMENT>]* )
    | <MSG_ARGUMENT>
9 <MSG_ARGUMENT> ::= <STRING_CONST> | <VARIABLE> | <NUMBER>

```

```

8 <TYPE> ::= <MESSAGE_LABEL>
8 <TIMING> ::= <NUMBER> [<TIME_DIMENSION>]
7 <OPERATOR_RESPONSE> ::= Y/N I (<VAR_ID_LIST>) [.] [Y/N]
    I <VAR_ID_LIST> [Y/N]
8 <VAR_ID_LIST> ::= <IDENTIFIER> [ , <IDENTIFIER> ] *
6 <KEYWORDED_DIAG> ::= [ OPERATOR MESSAGE : ] <DIAG_KEYWORD> [ , <DIAG_KEYWORD> ] *
7 <DIAG_KEYWORD> ::= [ AFFECTED ] COMPONENT = <AFFECTED_COMPONENTS>
    I [ OTHER ] PARAMETER = <OTHER_PARAMETERS>
    I TYPE = <TYPE>
    I TIME = <TIMING>
    I RESPONSE = <OPERATOR_RESPONSE>
4 <MESSAGE_DEFINITION> ::= MESSAGE <MESSAGE_LABEL> [ : ]
    [ ALIAS = <SYNONYM> . ] [ TEXT = ] <MESSAGE_TEXT> I
5 <MESSAGE_LABEL> ::= <LABEL>
5 <SYNONYM> ::= <IDENTIFIER>
5 <MESSAGE_TEXT> ::= (<TEXT_ELEM> [ , <TEXT_ELEM> ] *)
    I <TEXT_ELEM> [ , <TEXT_ELEM> ] *
6 <TEXT_ELEM> ::= <CHAR_STRING> I COMPONENT <ELEM_INDEX>
    I PARAMETER <ELEM_INDEX> I $TEST I SKIP <ELEM_INDEX>
7 <ELEM_INDEX> ::= [ (<UNSIGNED_INTEGER> ) ]

```



```

3 <UUT_SPEC> ::= <UUT_COMPONENT_FAILURE> [<UUT_COMPONENT_FAILURE>]*
    | <UUT_CONNECTION_POINT> [<UUT_CONNECTION_POINT>]*
4 <UUT_COMPONENT_FAILURE> ::= COMP_FAIL [<COMP_FAIL_SEQ#>] [:] <COMPONENT>
    [, <COMP_FAIL_KEYWD>]* ;
5 <COMP_FAIL_SEQ#> ::= <ENTRY_SEQ#>
6 <ENTRY_SEQ#> ::= <UNSIGNED_INTEGER>
5 <COMP_FAIL_KEYWD> ::= ALIAS = <SYNONYM>
    | FAILURE [FUNCTION] = <FAILURE_FUNCTION>
    | PARAMETER = <PARAM_LIST> | INDEX = <FAILURE_INDEX>
    | PROTECTION = <PROTECTION> | <COMMENTS>
6 <FAILURE_FUNCTION> ::= <FUNCTION_ID>
6 <PARAM_LIST> ::= (<PARAM_NAME> [, <PARAM_NAME>]* ) | <PARAM_NAME>
7 <PARAM_NAME> ::= <IDENTIFIER>
6 <FAILURE_INDEX> ::= <INTEGER>
6 <PROTECTION> ::= (<COMP_FAIL_ID> [<COMP_FAIL_ID>]* ) | <COMP_FAIL_ID>
6 <COMP_FAIL_ID> ::= <COMP_FAIL_SEQ#> | <COMPONENT> |
    <FAILURE_FUNCTION> (<COMPONENT>)
6 <COMMENTS> ::= COMMENT = <CHAR_STRING>
4 <UUT_CONNECTION_POINT> ::= UUT_POINT [<ENTRY_SEQ#>] [:] <UUT_POINT_ID>
    [, <UUT_POINT_KEYWD>]* ;
5 <UUT_POINT_ID> ::= <IDENTIFIER>

```

```

5 <UUT_POINT_KEYWD> ::= ALIAS = <SYNONYM>
    I CONNECTOR = <UUT_CONNECTOR>
    I LIMIT = <PROTECTIVE_LIMITS> I <COMMENTS>

6 <UUT_CONNECTOR> ::= (<CONN_TYPE> [, <CONN_POINT>]) I <CONN_TYPE>
    7 <CONN_TYPE> ::= <IDENTIFIER>
    7 <CONN_POINT> ::= <IDENTIFIER>

6 <PROTECTIVE_LIMITS> ::= ([<DIMENSION>] [, [<MAX_LIMIT>] [, [<MIN_LIMIT>]
    [, <REFERENCE_POINT>]]) I
    I <DIMENSION>

    7 <MAX_LIMIT> ::= <NUMBER>
    7 <MIN_LIMIT> ::= <NUMBER>
    7 <REFERENCE_POINT> ::= <UUT_POINT_ID>

3 <ATE_SPEC> ::= <ATE_FUNCTION> [<ATE_FUNCTION>]*
    I <ATE_CONNECTION_POINT> [<ATE_CONNECTION_POINT>]*

4 <ATE_FUNCTION> ::= FUNCTION [<ENTRY_SEQ#>] [:] <FUNCTION_ID>
    [, <FUNCTION_KEYWD>]* I

5 <FUNCTION_KEYWD> ::= ALIAS = <SYNONYM>
    I [FUNCTION] TYPE = <FUNCTION_TYPE>
    I #PINS = <UNSIGNED_INTEGER>
    I PARAMETER = <PARAM> [, PARAMETER = <PARAM>]*

```

```

1 VALUE [RETURNED] = <VALUES_RETURNED>
1 COOPERATION = <COOP_FUNCTIONS> I <COMMENTS>

6 <FUNCTION_TYPE> ::= S I M I F I E I C
6 <PARM> ::= (<PARM_NAME> [ , [<PARM_TYPE>] [ , [LIMIT=]<PARM_LIMITS>] ] )
1 <PARM_NAME>

7 <PARM_TYPE> ::= S I T
7 <PARM_LIMITS> ::= [ <DIMENSION> ] [ , [<MAX_LIMIT>] [ , <MIN_LIMIT>] ] )
1 <DIMENSION>

6 <VALUES_RETURNED> ::= <CHAR_STRING>
6 <COOP_FUNCTIONS> ::= (<FUNCTION_ID> [ , <FUNCTION_ID> ] * ) I <FUNCTION_ID>
4 <ATE_CONNECTION_POINT> ::= ATE_POINT [ <ENTRY_SEQH> ] [ : ] <ATE_POINT_ID>
[ , <ATE_POINT_KEYWD> ] * ;

5 <ATE_POINT_ID> ::= <IDENTIFIER>
5 <ATE_POINT_KEYWD> ::= ALIAS = <SYNONYM>
1 UUT_POINT = <UUT_POINTS> I <COMMENTS>
6 <UUT_POINTS> ::= (<UUT_POINT_ID> [ , <UUT_POINT_ID> ] * )
1 <UUT_POINT_ID>

```


APPENDIX C

FITS SYSTEM TAPE CONTENTS

Standard tape label: FITS
Tape : 2400', 1/2", 800bpi
Record size : 80
Block size : 2400

| NO | DATA SET NAME | DESCRIPTION |
|----|---------------|---|
| 1 | NAP2.JCL | JCL to execute the NAP2 circuit analysis program on IBM/370 computers |
| 2 | FITS.JCL | JCL to execute the FITS system |
| 3 | NAP2.OBJ | object code of NAP2 program |
| 4 | FITS.OBJ | object code of FITS |
| 5 | NAP2.SRC | FORTTRAN source code of NAP2 as supplied by the Technical University of Denmark |
| 6 | FITS.SRC | FORTTRAN source code of the FITS system |